Night Byte

Generated by Doxygen 1.8.20

Chapter 1

Night Byte Game

Running the game

See `the Engine repository` for build and run instructions.

Chapter 2

The server_decoration protocol

2.1 Interfaces

- org_kde_kwin_server_decoration_manager - Server side window decoration manager
- org_kde_kwin_server_decoration -

2.2 Copyright

2.3 org_kde_kwin_server_decoration_manager

2.3.1 Description

This interface allows to coordinate whether the server should create a server-side window decoration around a wl_surface representing a shell surface (wl_shell_surface or similar). By announcing support for this interface the server indicates that it supports server side decorations.

2.3.2 API

See The org_kde_kwin_server_decoration_manager interface.

2.4 org_kde_kwin_server_decoration

2.4.1 API

See The org_kde_kwin_server_decoration interface.

Chapter 3

The pointer_constraints_unstable_v1 protocol

protocol for constraining pointer motions

3.1 Description

This protocol specifies a set of interfaces used for adding constraints to the motion of a pointer. Possible constraints include confining pointer motions to a given region, or locking it to its current position.

In order to constrain the pointer, a client must first bind the global interface "wp_pointer_constraints" which, if a compositor supports pointer constraints, is exposed by the registry. Using the bound global object, the client uses the request that corresponds to the type of constraint it wants to make. See wp_pointer_constraints for more details.

Warning! The protocol described in this file is experimental and backward incompatible changes may be made. Backward compatible changes may be added together with the corresponding interface version bump. Backward incompatible changes are done by bumping the version number in the protocol and interface names and resetting the interface version. Once the protocol is to be declared stable, the 'z' prefix and the version number in the protocol and interface names are removed and the interface version number is reset.

3.2 Interfaces

- zwp_pointer_constraints_v1 - constrain the movement of a pointer
- zwp_locked_pointer_v1 - receive relative pointer motion events
- zwp_confined_pointer_v1 - confined pointer object

3.3 Copyright

3.4 zwp_pointer_constraints_v1

3.4.1 Description

The global interface exposing pointer constraining functionality. It exposes two requests: lock_pointer for locking the pointer to its position, and confine_pointer for locking the pointer to a region.

The lock_pointer and confine_pointer requests create the objects wp_locked_pointer and wp_confined_pointer respectively, and the client can use these objects to interact with the lock.

For any surface, only one lock or confinement may be active across all wl_pointer objects of the same seat. If a lock or confinement is requested when another lock or confinement is active or requested on the same surface and with any of the wl_pointer objects of the same seat, an 'already_constrained' error will be raised.

3.4.2 API

See The zwp_pointer_constraints_v1 interface.

3.5 zwp_locked_pointer_v1

3.5.1 Description

The wp_locked_pointer interface represents a locked pointer state.

While the lock of this object is active, the wl_pointer objects of the associated seat will not emit any wl_pointer.motion events.

This object will send the event 'locked' when the lock is activated. Whenever the lock is activated, it is guaranteed that the locked surface will already have received pointer focus and that the pointer will be within the region passed to the request creating this object.

To unlock the pointer, send the destroy request. This will also destroy the wp_locked_pointer object.

If the compositor decides to unlock the pointer the unlocked event is sent. See wp_locked_pointer.unlock for details.

When unlocking, the compositor may warp the cursor position to the set cursor position hint. If it does, it will not result in any relative motion events emitted via wp_relative_pointer.

If the surface the lock was requested on is destroyed and the lock is not yet activated, the wp_locked_pointer object is now defunct and must be destroyed.

3.5.2 API

See The zwp_locked_pointer_v1 interface.

3.6 zwp_confined_pointer_v1

3.6.1 Description

The wp_confined_pointer interface represents a confined pointer state.

This object will send the event 'confined' when the confinement is activated. Whenever the confinement is activated, it is guaranteed that the surface the pointer is confined to will already have received pointer focus and that the pointer will be within the region passed to the request creating this object. It is up to the compositor to decide whether this requires some user interaction and if the pointer will warp to within the passed region if outside.

To unconfine the pointer, send the destroy request. This will also destroy the wp_confined_pointer object.

If the compositor decides to unconfine the pointer the unconfined event is sent. The wp_confined_pointer object is at this point defunct and should be destroyed.

3.6.2 API

See The zwp_confined_pointer_v1 interface.

Chapter 4

The relative_pointer_unstable_v1 protocol

protocol for relative pointer motion events

4.1 Description

This protocol specifies a set of interfaces used for making clients able to receive relative pointer events not obstructed by barriers (such as the monitor edge or other pointer barriers).

To start receiving relative pointer events, a client must first bind the global interface "wp_relative_pointer_manager" which, if a compositor supports relative pointer motion events, is exposed by the registry. After having created the relative pointer manager proxy object, the client uses it to create the actual relative pointer object using the "get_relative_pointer" request given a wl_pointer. The relative pointer motion events will then, when applicable, be transmitted via the proxy of the newly created relative pointer object. See the documentation of the relative pointer interface for more details.

Warning! The protocol described in this file is experimental and backward incompatible changes may be made. Backward compatible changes may be added together with the corresponding interface version bump. Backward incompatible changes are done by bumping the version number in the protocol and interface names and resetting the interface version. Once the protocol is to be declared stable, the 'z' prefix and the version number in the protocol and interface names are removed and the interface version number is reset.

4.2 Interfaces

- zwp_relative_pointer_manager_v1 - get relative pointer objects
- zwp_relative_pointer_v1 - relative pointer object

4.3 Copyright

4.4 zwp_relative_pointer_manager_v1

4.4.1 Description

A global interface used for getting the relative pointer object for a given pointer.

4.4.2 API

See The zwp_relative_pointer_manager_v1 interface.

4.5 zwp_relative_pointer_v1

4.5.1 Description

A wp_relative_pointer object is an extension to the wl_pointer interface used for emitting relative pointer events. It shares the same focus as wl_pointer objects of the same seat and will only emit events when it has focus.

4.5.2 API

See The zwp_relative_pointer_v1 interface.

Chapter 5

The wayland protocol

5.1 Interfaces

- wl_display - core global object
- wl_registry - global registry object
- wl_callback - callback object
- wl_compositor - the compositor singleton
- wl_shm_pool - a shared memory pool
- wl_shm - shared memory support
- wl_buffer - content for a wl_surface
- wl_data_offer - offer to transfer data
- wl_data_source - offer to transfer data
- wl_data_device - data transfer device
- wl_data_device_manager - data transfer interface
- wl_shell - create desktop-style surfaces
- wl_shell_surface - desktop-style metadata interface
- wl_surface - an onscreen surface
- wl_seat - group of input devices
- wl_pointer - pointer input device
- wl_keyboard - keyboard input device
- wl_touch - touchscreen input device
- wl_output - compositor output region
- wl_region - region interface
- wl_subcompositor - sub-surface compositing
- wl_subsurface - sub-surface interface to a wl_surface

5.2 Copyright

5.3 wl_display

5.3.1 Description

The core global object. This is a special singleton object. It is used for internal Wayland protocol features.

5.3.2 API

See The wl_display interface.

5.4 wl_registry

5.4.1 Description

The singleton global registry object. The server has a number of global objects that are available to all clients. These objects typically represent an actual object in the server (for example, an input device) or they are singleton objects that provide extension functionality.

When a client creates a registry object, the registry object will emit a global event for each global currently in the registry. Globals come and go as a result of device or monitor hotplugs, reconfiguration or other events, and the registry will send out global and global_remove events to keep the client up to date with the changes. To mark the end of the initial burst of events, the client can use the wl_display.sync request immediately after calling wl_↩ display.get_registry.

A client can bind to a global object by using the bind request. This creates a client-side handle that lets the object emit events to the client and lets the client invoke requests on the object.

5.4.2 API

See The wl_registry interface.

5.5 wl_callback

5.5.1 Description

Clients can handle the 'done' event to get notified when the related request is done.

5.5.2 API

See The wl_callback interface.

5.6 wl_compositor

5.6.1 Description

A compositor. This object is a singleton global. The compositor is in charge of combining the contents of multiple surfaces into one displayable output.

5.6.2 API

See The wl_compositor interface.

5.7 wl_shm_pool

5.7.1 Description

The wl_shm_pool object encapsulates a piece of memory shared between the compositor and client. Through the wl_shm_pool object, the client can allocate shared memory wl_buffer objects. All objects created through the same pool share the same underlying mapped memory. Reusing the mapped memory avoids the setup/teardown overhead and is useful when interactively resizing a surface or for many small buffers.

5.7.2 API

See The wl_shm_pool interface.

5.8 wl_shm

5.8.1 Description

A singleton global object that provides support for shared memory.

Clients can create wl_shm_pool objects using the create_pool request.

At connection setup time, the wl_shm object emits one or more format events to inform clients about the valid pixel formats that can be used for buffers.

5.8.2 API

See The wl_shm interface.

5.9 wl_buffer

5.9.1 Description

A buffer provides the content for a wl_surface. Buffers are created through factory interfaces such as wl_drm, wl↩ _shm or similar. It has a width and a height and can be attached to a wl_surface, but the mechanism by which a client provides and updates the contents is defined by the buffer factory interface.

5.9.2 API

See The wl_buffer interface.

5.10 wl_data_offer

5.10.1 Description

A wl_data_offer represents a piece of data offered for transfer by another client (the source client). It is used by the copy-and-paste and drag-and-drop mechanisms. The offer describes the different mime types that the data can be converted to and provides the mechanism for transferring the data directly from the source client.

5.10.2 API

See The wl_data_offer interface.

5.11 wl_data_source

5.11.1 Description

The wl_data_source object is the source side of a wl_data_offer. It is created by the source client in a data transfer and provides a way to describe the offered data and a way to respond to requests to transfer the data.

5.11.2 API

See The wl_data_source interface.

5.12 wl_data_device

5.12.1 Description

There is one wl_data_device per seat which can be obtained from the global wl_data_device_manager singleton.

A wl_data_device provides access to inter-client data transfer mechanisms such as copy-and-paste and drag-and-drop.

5.12.2 API

See The wl_data_device interface.

5.13 wl_data_device_manager

5.13.1 Description

The wl_data_device_manager is a singleton global object that provides access to inter-client data transfer mechanisms such as copy-and-paste and drag-and-drop. These mechanisms are tied to a wl_seat and this interface lets a client get a wl_data_device corresponding to a wl_seat.

Depending on the version bound, the objects created from the bound wl_data_device_manager object will have different requirements for functioning properly. See wl_data_source.set_actions, wl_data_offer.accept and wl_↩ data_offer.finish for details.

5.13.2 API

See The wl_data_device_manager interface.

5.14 wl_shell

5.14.1 Description

This interface is implemented by servers that provide desktop-style user interfaces.

It allows clients to associate a wl_shell_surface with a basic surface.

5.14.2 API

See The wl_shell interface.

5.15 wl_shell_surface

5.15.1 Description

An interface that may be implemented by a wl_surface, for implementations that provide a desktop-style user interface.

It provides requests to treat surfaces like toplevel, fullscreen or popup windows, move, resize or maximize them, associate metadata like title and class, etc.

On the server side the object is automatically destroyed when the related wl_surface is destroyed. On the client side, wl_shell_surface_destroy() must be called before destroying the wl_surface object.

5.15.2 API

See The wl_shell_surface interface.

5.16 wl_surface

5.16.1 Description

A surface is a rectangular area that is displayed on the screen. It has a location, size and pixel contents.

The size of a surface (and relative positions on it) is described in surface-local coordinates, which may differ from the buffer coordinates of the pixel content, in case a buffer_transform or a buffer_scale is used.

A surface without a "role" is fairly useless: a compositor does not know where, when or how to present it. The role is the purpose of a wl_surface. Examples of roles are a cursor for a pointer (as set by wl_pointer.set_cursor), a drag icon (wl_data_device.start_drag), a sub-surface (wl_subcompositor.get_subsurface), and a window as defined by a shell protocol (e.g. wl_shell.get_shell_surface).

A surface can have only one role at a time. Initially a wl_surface does not have a role. Once a wl_surface is given a role, it is set permanently for the whole lifetime of the wl_surface object. Giving the current role again is allowed, unless explicitly forbidden by the relevant interface specification.

Surface roles are given by requests in other interfaces such as wl_pointer.set_cursor. The request should explicitly mention that this request gives a role to a wl_surface. Often, this request also creates a new protocol object that represents the role and adds additional functionality to wl_surface. When a client wants to destroy a wl_surface, they must destroy this 'role object' before the wl_surface.

Destroying the role object does not remove the role from the wl_surface, but it may stop the wl_surface from "playing the role". For instance, if a wl_subsurface object is destroyed, the wl_surface it was created for will be unmapped and forget its position and z-order. It is allowed to create a wl_subsurface for the same wl_surface again, but it is not allowed to use the wl_surface as a cursor (cursor is a different role than sub-surface, and role switching is not allowed).

5.16.2 API

See The wl_surface interface.

5.17 wl_seat

5.17.1 Description

A seat is a group of keyboards, pointer and touch devices. This object is published as a global during start up, or when such a device is hot plugged. A seat typically has a pointer and maintains a keyboard focus and a pointer focus.

5.17.2 API

See The wl_seat interface.

5.18 wl_pointer

5.18.1 Description

The wl_pointer interface represents one or more input devices, such as mice, which control the pointer location and pointer_focus of a seat.

The wl_pointer interface generates motion, enter and leave events for the surfaces that the pointer is located over, and button and axis events for button presses, button releases and scrolling.

5.18.2 API

See The wl_pointer interface.

5.19 wl_keyboard

5.19.1 Description

The wl_keyboard interface represents one or more keyboards associated with a seat.

5.19.2 API

See The wl_keyboard interface.

5.20 wl_touch

5.20.1 Description

The wl_touch interface represents a touchscreen associated with a seat.

Touch interactions can consist of one or more contacts. For each contact, a series of events is generated, starting with a down event, followed by zero or more motion events, and ending with an up event. Events relating to the same contact point can be identified by the ID of the sequence.

5.20.2 API

See The wl_touch interface.

5.21 wl_output

5.21.1 Description

An output describes part of the compositor geometry. The compositor works in the 'compositor coordinate system' and an output corresponds to a rectangular area in that space that is actually visible. This typically corresponds to a monitor that displays part of the compositor space. This object is published as global during start up, or when a monitor is hotplugged.

5.21.2 API

See The wl_output interface.

5.22 wl_region

5.22.1 Description

A region object describes an area.

Region objects are used to describe the opaque and input regions of a surface.

5.22.2 API

See The wl_region interface.

5.23 wl_subcompositor

5.23.1 Description

The global interface exposing sub-surface compositing capabilities. A wl_surface, that has sub-surfaces associated, is called the parent surface. Sub-surfaces can be arbitrarily nested and create a tree of sub-surfaces.

The root surface in a tree of sub-surfaces is the main surface. The main surface cannot be a sub-surface, because sub-surfaces must always have a parent.

A main surface with its sub-surfaces forms a (compound) window. For window management purposes, this set of wl_surface objects is to be considered as a single window, and it should also behave as such.

The aim of sub-surfaces is to offload some of the compositing work within a window from clients to the compositor. A prime example is a video player with decorations and video in separate wl_surface objects. This should allow the compositor to pass YUV video buffer processing to dedicated overlay hardware when possible.

5.23.2 API

See The wl_subcompositor interface.

5.24 wl_subsurface

5.24.1 Description

An additional interface to a wl_surface object, which has been made a sub-surface. A sub-surface has one parent surface. A sub-surface's size and position are not limited to that of the parent. Particularly, a sub-surface is not automatically clipped to its parent's area.

A sub-surface becomes mapped, when a non-NULL wl_buffer is applied and the parent surface is mapped. The order of which one happens first is irrelevant. A sub-surface is hidden if the parent becomes hidden, or if a NULL wl_buffer is applied. These rules apply recursively through the tree of surfaces.

The behaviour of a wl_surface.commit request on a sub-surface depends on the sub-surface's mode. The possible modes are synchronized and desynchronized, see methods wl_subsurface.set_sync and wl_subsurface.set_↩ desync. Synchronized mode caches the wl_surface state to be applied when the parent's state gets applied, and desynchronized mode applies the pending wl_surface state directly. A sub-surface is initially in the synchronized mode.

Sub-surfaces have also other kind of state, which is managed by wl_subsurface requests, as opposed to wl_surface requests. This state includes the sub-surface position relative to the parent surface (wl_subsurface.set_position), and the stacking order of the parent and its sub-surfaces (wl_subsurface.place_above and .place_below). This state is applied when the parent surface's wl_surface state is applied, regardless of the sub-surface's mode. As the exception, set_sync and set_desync are effective immediately.

The main surface can be thought to be always in desynchronized mode, since it does not have a parent in the sub-surfaces sense.

Even if a sub-surface is in desynchronized mode, it will behave as in synchronized mode, if its parent surface behaves as in synchronized mode. This rule is applied recursively throughout the tree of surfaces. This means, that one can set a sub-surface into synchronized mode, and then assume that all its child and grand-child sub-surfaces are synchronized, too, without explicitly setting them.

If the wl_surface associated with the wl_subsurface is destroyed, the wl_subsurface object becomes inert. Note, that destroying either object takes effect immediately. If you need to synchronize the removal of a sub-surface to the parent surface update, unmap the sub-surface first by attaching a NULL wl_buffer, update parent, and then destroy the sub-surface.

If the parent wl_surface object is destroyed, the sub-surface is unmapped.

5.24.2 API

See The wl_subsurface interface.

Chapter 6

The xdg_decoration_unstable_v1 protocol

6.1 Interfaces

- zxdg_decoration_manager_v1 - window decoration manager
- zxdg_toplevel_decoration_v1 - decoration object for a toplevel surface

6.2 Copyright

6.3 zxdg_decoration_manager_v1

6.3.1 Description

This interface allows a compositor to announce support for server-side decorations.

A window decoration is a set of window controls as deemed appropriate by the party managing them, such as user interface components used to move, resize and change a window's state.

A client can use this protocol to request being decorated by a supporting compositor.

If compositor and client do not negotiate the use of a server-side decoration using this protocol, clients continue to self-decorate as they see fit.

Warning! The protocol described in this file is experimental and backward incompatible changes may be made. Backward compatible changes may be added together with the corresponding interface version bump. Backward incompatible changes are done by bumping the version number in the protocol and interface names and resetting the interface version. Once the protocol is to be declared stable, the 'z' prefix and the version number in the protocol and interface names are removed and the interface version number is reset.

6.3.2 API

See The zxdg_decoration_manager_v1 interface.

6.4 zxdg_toplevel_decoration_v1

6.4.1 Description

The decoration object allows the compositor to toggle server-side window decorations for a toplevel surface. The client can request to switch to another mode.

The xdg_toplevel_decoration object must be destroyed before its xdg_toplevel.

6.4.2 API

See The zxdg_toplevel_decoration_v1 interface.

Chapter 7

The xdg_shell protocol

7.1 Interfaces

- xdg_wm_base - create desktop-style surfaces
- xdg_positioner - child surface positioner
- xdg_surface - desktop user interface surface base interface
- xdg_toplevel - toplevel surface
- xdg_popup - short-lived, popup surfaces for menus

7.2 Copyright

7.3 xdg_wm_base

7.3.1 Description

The xdg_wm_base interface is exposed as a global object enabling clients to turn their wl_surfaces into windows in a desktop environment. It defines the basic functionality needed for clients and the compositor to create windows that can be dragged, resized, maximized, etc, as well as creating transient windows such as popup menus.

7.3.2 API

See The xdg_wm_base interface.

7.4 xdg_positioner

7.4.1 Description

The xdg_positioner provides a collection of rules for the placement of a child surface relative to a parent surface. Rules can be defined to ensure the child surface remains within the visible area's borders, and to specify how the child surface changes its position, such as sliding along an axis, or flipping around a rectangle. These positioner-created rules are constrained by the requirement that a child surface must intersect with or be at least partially adjacent to its parent surface.

See the various requests for details about possible rules.

At the time of the request, the compositor makes a copy of the rules specified by the xdg_positioner. Thus, after the request is complete the xdg_positioner object can be destroyed or reused; further changes to the object will have no effect on previous usages.

For an xdg_positioner object to be considered complete, it must have a non-zero size set by set_size, and a non-zero anchor rectangle set by set_anchor_rect. Passing an incomplete xdg_positioner object when positioning a surface raises an error.

7.4.2 API

See The xdg_positioner interface.

7.5 xdg_surface

7.5.1 Description

An interface that may be implemented by a wl_surface, for implementations that provide a desktop-style user interface.

It provides a base set of functionality required to construct user interface elements requiring management by the compositor, such as toplevel windows, menus, etc. The types of functionality are split into xdg_surface roles.

Creating an xdg_surface does not set the role for a wl_surface. In order to map an xdg_surface, the client must create a role-specific object using, e.g., get_toplevel, get_popup. The wl_surface for any given xdg_surface can have at most one role, and may not be assigned any role not based on xdg_surface.

A role must be assigned before any other requests are made to the xdg_surface object.

The client must call wl_surface.commit on the corresponding wl_surface for the xdg_surface state to take effect.

Creating an xdg_surface from a wl_surface which has a buffer attached or committed is a client error, and any attempts by a client to attach or manipulate a buffer prior to the first xdg_surface.configure call must also be treated as errors.

Mapping an xdg_surface-based role surface is defined as making it possible for the surface to be shown by the compositor. Note that a mapped surface is not guaranteed to be visible once it is mapped.

For an xdg_surface to be mapped by the compositor, the following conditions must be met: (1) the client has assigned an xdg_surface-based role to the surface (2) the client has set and committed the xdg_surface state and the role-dependent state to the surface (3) the client has committed a buffer to the surface

A newly-unmapped surface is considered to have met condition (1) out of the 3 required conditions for mapping a surface if its role surface has not been destroyed.

7.5.2 API

See The xdg_surface interface.

7.6 xdg_toplevel

7.6.1 Description

This interface defines an xdg_surface role which allows a surface to, among other things, set window-like properties such as maximize, fullscreen, and minimize, set application-specific metadata like title and id, and well as trigger user interactive operations such as interactive resize and move.

Unmapping an xdg_toplevel means that the surface cannot be shown by the compositor until it is explicitly mapped again. All active operations (e.g., move, resize) are canceled and all attributes (e.g. title, state, stacking, ...) are discarded for an xdg_toplevel surface when it is unmapped.

Attaching a null buffer to a toplevel unmaps the surface.

7.6.2 API

See The xdg_toplevel interface.

7.7 xdg_popup

7.7.1 Description

A popup surface is a short-lived, temporary surface. It can be used to implement for example menus, popovers, tooltips and other similar user interface concepts.

A popup can be made to take an explicit grab. See xdg_popup.grab for details.

When the popup is dismissed, a popup_done event will be sent out, and at the same time the surface will be unmapped. See the xdg_popup.popup_done event for details.

Explicitly destroying the xdg_popup object will also dismiss the popup and unmap the surface. Clients that want to dismiss the popup when another surface of their own is clicked should dismiss the popup using the destroy request.

The parent surface must have either the xdg_toplevel or xdg_popup surface role.

A newly created xdg_popup will be stacked on top of all previously created xdg_popup surfaces associated with the same xdg_toplevel.

The parent of an xdg_popup must be mapped (see the xdg_surface description) before the xdg_popup itself.

The x and y arguments passed when creating the popup object specify where the top left of the popup should be placed, relative to the local surface coordinates of the parent surface. See xdg_surface.get_popup. An xdg_popup must intersect with or be at least partially adjacent to its parent surface.

The client must call wl_surface.commit on the corresponding wl_surface for the xdg_popup state to take effect.

7.7.2 API

See The xdg_popup interface.

Chapter 8

The xdg_shell_unstable_v6 protocol

8.1 Interfaces

- zxdg_shell_v6 - create desktop-style surfaces
- zxdg_positioner_v6 - child surface positioner
- zxdg_surface_v6 - desktop user interface surface base interface
- zxdg_toplevel_v6 - toplevel surface
- zxdg_popup_v6 - short-lived, popup surfaces for menus

8.2 Copyright

8.3 zxdg_shell_v6

8.3.1 Description

xdg_shell allows clients to turn a wl_surface into a "real window" which can be dragged, resized, stacked, and moved around by the user. Everything about this interface is suited towards traditional desktop environments.

8.3.2 API

See The zxdg_shell_v6 interface.

8.4 zxdg_positioner_v6

8.4.1 Description

The xdg_positioner provides a collection of rules for the placement of a child surface relative to a parent surface. Rules can be defined to ensure the child surface remains within the visible area's borders, and to specify how the child surface changes its position, such as sliding along an axis, or flipping around a rectangle. These positioner-created rules are constrained by the requirement that a child surface must intersect with or be at least partially adjacent to its parent surface.

See the various requests for details about possible rules.

At the time of the request, the compositor makes a copy of the rules specified by the xdg_positioner. Thus, after the request is complete the xdg_positioner object can be destroyed or reused; further changes to the object will have no effect on previous usages.

For an xdg_positioner object to be considered complete, it must have a non-zero size set by set_size, and a non-zero anchor rectangle set by set_anchor_rect. Passing an incomplete xdg_positioner object when positioning a surface raises an error.

8.4.2 API

See The zxdg_positioner_v6 interface.

8.5 zxdg_surface_v6

8.5.1 Description

An interface that may be implemented by a wl_surface, for implementations that provide a desktop-style user interface.

It provides a base set of functionality required to construct user interface elements requiring management by the compositor, such as toplevel windows, menus, etc. The types of functionality are split into xdg_surface roles.

Creating an xdg_surface does not set the role for a wl_surface. In order to map an xdg_surface, the client must create a role-specific object using, e.g., get_toplevel, get_popup. The wl_surface for any given xdg_surface can have at most one role, and may not be assigned any role not based on xdg_surface.

A role must be assigned before any other requests are made to the xdg_surface object.

The client must call wl_surface.commit on the corresponding wl_surface for the xdg_surface state to take effect.

Creating an xdg_surface from a wl_surface which has a buffer attached or committed is a client error, and any attempts by a client to attach or manipulate a buffer prior to the first xdg_surface.configure call must also be treated as errors.

For a surface to be mapped by the compositor, the following conditions must be met: (1) the client has assigned a xdg_surface based role to the surface, (2) the client has set and committed the xdg_surface state and the role dependent state to the surface and (3) the client has committed a buffer to the surface.

8.5.2 API

See The zxdg_surface_v6 interface.

8.6 zxdg_toplevel_v6

8.6.1 Description

This interface defines an xdg_surface role which allows a surface to, among other things, set window-like properties such as maximize, fullscreen, and minimize, set application-specific metadata like title and id, and well as trigger user interactive operations such as interactive resize and move.

8.6.2 API

See The zxdg_toplevel_v6 interface.

8.7 zxdg_popup_v6

8.7.1 Description

A popup surface is a short-lived, temporary surface. It can be used to implement for example menus, popovers, tooltips and other similar user interface concepts.

A popup can be made to take an explicit grab. See xdg_popup.grab for details.

When the popup is dismissed, a popup_done event will be sent out, and at the same time the surface will be unmapped. See the xdg_popup.popup_done event for details.

Explicitly destroying the xdg_popup object will also dismiss the popup and unmap the surface. Clients that want to dismiss the popup when another surface of their own is clicked should dismiss the popup using the destroy request.

The parent surface must have either the xdg_toplevel or xdg_popup surface role.

A newly created xdg_popup will be stacked on top of all previously created xdg_popup surfaces associated with the same xdg_toplevel.

The parent of an xdg_popup must be mapped (see the xdg_surface description) before the xdg_popup itself.

The x and y arguments passed when creating the popup object specify where the top left of the popup should be placed, relative to the local surface coordinates of the parent surface. See xdg_surface.get_popup. An xdg_popup must intersect with or be at least partially adjacent to its parent surface.

The client must call wl_surface.commit on the corresponding wl_surface for the xdg_popup state to take effect.

8.7.2 API

See The zxdg_popup_v6 interface.

Chapter 9

Module Index

9.1 Modules

Here is a list of all modules:

Chapter 10

Namespace Index

10.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 11

Hierarchical Index

11.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 12

Class Index

12.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 13

File Index

13.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 14

Module Documentation

14.1 The org_kde_kwin_server_decoration_manager interface

Classes

- struct org_kde_kwin_server_decoration_manager_listener

Macros

- #define **ORG_KDE_KWIN_SERVER_DECORATION_MANAGER_DEFAULT_MODE_SINCE_VERSION** 1
- #define **ORG_KDE_KWIN_SERVER_DECORATION_MANAGER_CREATE_SINCE_VERSION** 1

Enumerations

- enum org_kde_kwin_server_decoration_manager_mode { ORG_KDE_KWIN_SERVER_DECORATION_MANAGER_MODE_N = 0, ORG_KDE_KWIN_SERVER_DECORATION_MANAGER_MODE_CLIENT = 1, ORG_KDE_KWIN_SERVER_DECORATI = 2 }

14.1.1 Detailed Description

This interface allows to coordinate whether the server should create a server-side window decoration around a wl_surface representing a shell surface (wl_shell_surface or similar). By announcing support for this interface the server indicates that it supports server side decorations.

14.1.2 Enumeration Type Documentation

14.1.2.1 org_kde_kwin_server_decoration_manager_mode

```
enum org_kde_kwin_server_decoration_manager_mode
```

Possible values to use in request_mode and the event mode.

Enumerator

ORG_KDE_KWIN_SERVER_DECORATION_MANAGER_MODE_NONE	Undecorated: The surface is not decorated at all, neither server nor client-side. An example is a popup surface which should not be decorated.
ORG_KDE_KWIN_SERVER_DECORATION_MANAGER_MODE_CLIENT	Client-side decoration: The decoration is part of the surface and the client.
ORG_KDE_KWIN_SERVER_DECORATION_MANAGER_MODE_SERVER	Server-side decoration: The server embeds the surface into a decoration frame.

14.2 The org_kde_kwin_server_decoration interface

Classes

- struct org_kde_kwin_server_decoration_listener

Macros

- #define **ORG_KDE_KWIN_SERVER_DECORATION_MODE_SINCE_VERSION** 1
- #define **ORG_KDE_KWIN_SERVER_DECORATION_RELEASE_SINCE_VERSION** 1
- #define **ORG_KDE_KWIN_SERVER_DECORATION_REQUEST_MODE_SINCE_VERSION** 1

Enumerations

- enum org_kde_kwin_server_decoration_mode { ORG_KDE_KWIN_SERVER_DECORATION_MODE_NONE = 0, ORG_KDE_KWIN_SERVER_DECORATION_MODE_CLIENT = 1, ORG_KDE_KWIN_SERVER_DECORATION_MODE_S = 2 }

14.2.1 Detailed Description

14.2.2 Enumeration Type Documentation

14.2.2.1 org_kde_kwin_server_decoration_mode

```
enum org_kde_kwin_server_decoration_mode
```

Possible values to use in request_mode and the event mode.

Enumerator

ORG_KDE_KWIN_SERVER_DECORATION_MO↩ DE_NONE	Undecorated: The surface is not decorated at all, neither server nor client-side. An example is a popup surface which should not be decorated.
ORG_KDE_KWIN_SERVER_DECORATION_MO↩ DE_CLIENT	Client-side decoration: The decoration is part of the surface and the client.
ORG_KDE_KWIN_SERVER_DECORATION_MO↩ DE_SERVER	Server-side decoration: The server embeds the surface into a decoration frame.

14.3 The zwp_pointer_constraints_v1 interface

Macros

- #define **ZWP_POINTER_CONSTRAINTS_V1_DESTROY_SINCE_VERSION** 1
- #define **ZWP_POINTER_CONSTRAINTS_V1_LOCK_POINTER_SINCE_VERSION** 1
- #define **ZWP_POINTER_CONSTRAINTS_V1_CONFINE_POINTER_SINCE_VERSION** 1

Enumerations

- enum zwp_pointer_constraints_v1_error { ZWP_POINTER_CONSTRAINTS_V1_ERROR_ALREADY_CONSTRAINED = 1 }
- enum zwp_pointer_constraints_v1_lifetime { **ZWP_POINTER_CONSTRAINTS_V1_LIFETIME_ONESHOT** = 1, **ZWP_POINTER_CONSTRAINTS_V1_LIFETIME_PERSISTENT** = 2 }

14.3.1 Detailed Description

The global interface exposing pointer constraining functionality. It exposes two requests: lock_pointer for locking the pointer to its position, and confine_pointer for locking the pointer to a region.

The lock_pointer and confine_pointer requests create the objects wp_locked_pointer and wp_confined_pointer respectively, and the client can use these objects to interact with the lock.

For any surface, only one lock or confinement may be active across all wl_pointer objects of the same seat. If a lock or confinement is requested when another lock or confinement is active or requested on the same surface and with any of the wl_pointer objects of the same seat, an 'already_constrained' error will be raised.

14.3.2 Enumeration Type Documentation

14.3.2.1 zwp_pointer_constraints_v1_error

```
enum zwp_pointer_constraints_v1_error
```

wp_pointer_constraints error values

These errors can be emitted in response to wp_pointer_constraints requests.

Enumerator

ZWP_POINTER_CONSTRAINTS_V1_ERROR_ALREADY_CONSTRAINED	pointer constraint already requested on that surface

14.3.2.2 zwp_pointer_constraints_v1_lifetime

enum zwp_pointer_constraints_v1_lifetime

the pointer constraint may reactivate

A persistent pointer constraint may again reactivate once it has been deactivated. See the corresponding deactivation event (wp_locked_pointer.unlocked and wp_confined_pointer.unconfined) for details.

14.4 The zwp_locked_pointer_v1 interface

Classes

- struct zwp_locked_pointer_v1_listener

Macros

- #define **ZWP_LOCKED_POINTER_V1_LOCKED_SINCE_VERSION** 1
- #define **ZWP_LOCKED_POINTER_V1_UNLOCKED_SINCE_VERSION** 1
- #define **ZWP_LOCKED_POINTER_V1_DESTROY_SINCE_VERSION** 1
- #define **ZWP_LOCKED_POINTER_V1_SET_CURSOR_POSITION_HINT_SINCE_VERSION** 1
- #define **ZWP_LOCKED_POINTER_V1_SET_REGION_SINCE_VERSION** 1

14.4.1 Detailed Description

The wp_locked_pointer interface represents a locked pointer state.

While the lock of this object is active, the wl_pointer objects of the associated seat will not emit any wl_pointer.motion events.

This object will send the event 'locked' when the lock is activated. Whenever the lock is activated, it is guaranteed that the locked surface will already have received pointer focus and that the pointer will be within the region passed to the request creating this object.

To unlock the pointer, send the destroy request. This will also destroy the wp_locked_pointer object.

If the compositor decides to unlock the pointer the unlocked event is sent. See wp_locked_pointer.unlock for details.

When unlocking, the compositor may warp the cursor position to the set cursor position hint. If it does, it will not result in any relative motion events emitted via wp_relative_pointer.

If the surface the lock was requested on is destroyed and the lock is not yet activated, the wp_locked_pointer object is now defunct and must be destroyed.

14.5 The zwp_confined_pointer_v1 interface

Classes

- struct zwp_confined_pointer_v1_listener

Macros

- #define **ZWP_CONFINED_POINTER_V1_CONFINED_SINCE_VERSION** 1
- #define **ZWP_CONFINED_POINTER_V1_UNCONFINED_SINCE_VERSION** 1
- #define **ZWP_CONFINED_POINTER_V1_DESTROY_SINCE_VERSION** 1
- #define **ZWP_CONFINED_POINTER_V1_SET_REGION_SINCE_VERSION** 1

14.5.1 Detailed Description

The wp_confined_pointer interface represents a confined pointer state.

This object will send the event 'confined' when the confinement is activated. Whenever the confinement is activated, it is guaranteed that the surface the pointer is confined to will already have received pointer focus and that the pointer will be within the region passed to the request creating this object. It is up to the compositor to decide whether this requires some user interaction and if the pointer will warp to within the passed region if outside.

To unconfine the pointer, send the destroy request. This will also destroy the wp_confined_pointer object.

If the compositor decides to unconfine the pointer the unconfined event is sent. The wp_confined_pointer object is at this point defunct and should be destroyed.

14.6 The zwp_relative_pointer_manager_v1 interface

Macros

- #define **ZWP_RELATIVE_POINTER_MANAGER_V1_DESTROY_SINCE_VERSION** 1
- #define **ZWP_RELATIVE_POINTER_MANAGER_V1_GET_RELATIVE_POINTER_SINCE_VERSION** 1

14.6.1 Detailed Description

A global interface used for getting the relative pointer object for a given pointer.

14.7 The zwp_relative_pointer_v1 interface

Classes

- struct zwp_relative_pointer_v1_listener

Macros

- #define **ZWP_RELATIVE_POINTER_V1_RELATIVE_MOTION_SINCE_VERSION** 1
- #define **ZWP_RELATIVE_POINTER_V1_DESTROY_SINCE_VERSION** 1

14.7.1 Detailed Description

A wp_relative_pointer object is an extension to the wl_pointer interface used for emitting relative pointer events. It shares the same focus as wl_pointer objects of the same seat and will only emit events when it has focus.

14.8 The wl_display interface

Classes

- struct wl_display_listener

Macros

- #define **WL_DISPLAY_ERROR_SINCE_VERSION** 1
- #define **WL_DISPLAY_DELETE_ID_SINCE_VERSION** 1
- #define **WL_DISPLAY_SYNC_SINCE_VERSION** 1
- #define **WL_DISPLAY_GET_REGISTRY_SINCE_VERSION** 1

Enumerations

- enum wl_display_error { WL_DISPLAY_ERROR_INVALID_OBJECT = 0, WL_DISPLAY_ERROR_INVALID_METHOD = 1, WL_DISPLAY_ERROR_NO_MEMORY = 2 }

14.8.1 Detailed Description

The core global object. This is a special singleton object. It is used for internal Wayland protocol features.

14.8.2 Enumeration Type Documentation

14.8.2.1 wl_display_error

`enum wl_display_error`

global error values

These errors are global and can be emitted in response to any server request.

Enumerator

WL_DISPLAY_ERROR_INVALID_OBJECT	server couldn't find object
WL_DISPLAY_ERROR_INVALID_METHOD	method doesn't exist on the specified interface
WL_DISPLAY_ERROR_NO_MEMORY	server is out of memory

14.9 The wl_registry interface

Classes

- struct wl_registry_listener

Macros

- #define **WL_REGISTRY_GLOBAL_SINCE_VERSION** 1
- #define **WL_REGISTRY_GLOBAL_REMOVE_SINCE_VERSION** 1
- #define **WL_REGISTRY_BIND_SINCE_VERSION** 1

14.9.1 Detailed Description

The singleton global registry object. The server has a number of global objects that are available to all clients. These objects typically represent an actual object in the server (for example, an input device) or they are singleton objects that provide extension functionality.

When a client creates a registry object, the registry object will emit a global event for each global currently in the registry. Globals come and go as a result of device or monitor hotplugs, reconfiguration or other events, and the registry will send out global and global_remove events to keep the client up to date with the changes. To mark the end of the initial burst of events, the client can use the wl_display.sync request immediately after calling wl_↩ display.get_registry.

A client can bind to a global object by using the bind request. This creates a client-side handle that lets the object emit events to the client and lets the client invoke requests on the object.

14.10 The wl_callback interface

Classes

- struct wl_callback_listener

Macros

- #define **WL_CALLBACK_DONE_SINCE_VERSION** 1

14.10.1 Detailed Description

Clients can handle the 'done' event to get notified when the related request is done.

14.11 The wl_compositor interface

Macros

- #define **WL_COMPOSITOR_CREATE_SURFACE_SINCE_VERSION** 1
- #define **WL_COMPOSITOR_CREATE_REGION_SINCE_VERSION** 1

14.11.1 Detailed Description

A compositor. This object is a singleton global. The compositor is in charge of combining the contents of multiple surfaces into one displayable output.

14.12 The wl_shm_pool interface

Macros

- #define **WL_SHM_POOL_CREATE_BUFFER_SINCE_VERSION** 1
- #define **WL_SHM_POOL_DESTROY_SINCE_VERSION** 1
- #define **WL_SHM_POOL_RESIZE_SINCE_VERSION** 1

14.12.1 Detailed Description

The wl_shm_pool object encapsulates a piece of memory shared between the compositor and client. Through the wl_shm_pool object, the client can allocate shared memory wl_buffer objects. All objects created through the same pool share the same underlying mapped memory. Reusing the mapped memory avoids the setup/teardown overhead and is useful when interactively resizing a surface or for many small buffers.

14.13 The wl_shm interface

Classes

- struct wl_shm_listener

Macros

- #define **WL_SHM_FORMAT_SINCE_VERSION** 1
- #define **WL_SHM_CREATE_POOL_SINCE_VERSION** 1

Enumerations

- enum wl_shm_error { WL_SHM_ERROR_INVALID_FORMAT = 0, WL_SHM_ERROR_INVALID_STRIDE = 1, WL_SHM_ERROR_INVALID_FD = 2 }
- enum wl_shm_format {
  WL_SHM_FORMAT_ARGB8888 = 0, WL_SHM_FORMAT_XRGB8888 = 1, WL_SHM_FORMAT_C8 = 0x20203843, WL_SHM_FORMAT_RGB332 = 0x38424752,
  WL_SHM_FORMAT_BGR233 = 0x38524742, WL_SHM_FORMAT_XRGB4444 = 0x32315258, WL_SHM_FORMAT_XBGR44 = 0x32314258, WL_SHM_FORMAT_RGBX4444 = 0x32315852,
  WL_SHM_FORMAT_BGRX4444 = 0x32315842, WL_SHM_FORMAT_ARGB4444 = 0x32315241,
  WL_SHM_FORMAT_ABGR4444 = 0x32314241, WL_SHM_FORMAT_RGBA4444 = 0x32314152,
  WL_SHM_FORMAT_BGRA4444 = 0x32314142, WL_SHM_FORMAT_XRGB1555 = 0x35315258,
  WL_SHM_FORMAT_XBGR1555 = 0x35314258, WL_SHM_FORMAT_RGBX5551 = 0x35315852,
  WL_SHM_FORMAT_BGRX5551 = 0x35315842, WL_SHM_FORMAT_ARGB1555 = 0x35315241,
  WL_SHM_FORMAT_ABGR1555 = 0x35314241, WL_SHM_FORMAT_RGBA5551 = 0x35314152,
  WL_SHM_FORMAT_BGRA5551 = 0x35314142, WL_SHM_FORMAT_RGB565 = 0x36314752, WL_SHM_FORMAT_BGR565 = 0x36314742, WL_SHM_FORMAT_RGB888 = 0x34324752,
  WL_SHM_FORMAT_BGR888 = 0x34324742, WL_SHM_FORMAT_XBGR8888 = 0x34324258, WL_SHM_FORMAT_RGBX88 = 0x34325852, WL_SHM_FORMAT_BGRX8888 = 0x34325842,
  WL_SHM_FORMAT_ABGR8888 = 0x34324241, WL_SHM_FORMAT_RGBA8888 = 0x34324152,
  WL_SHM_FORMAT_BGRA8888 = 0x34324142, WL_SHM_FORMAT_XRGB2101010 = 0x30335258,
  WL_SHM_FORMAT_XBGR2101010 = 0x30334258, WL_SHM_FORMAT_RGBX1010102 = 0x30335852,
  WL_SHM_FORMAT_BGRX1010102 = 0x30335842, WL_SHM_FORMAT_ARGB2101010 = 0x30335241,
  WL_SHM_FORMAT_ABGR2101010 = 0x30334241, WL_SHM_FORMAT_RGBA1010102 = 0x30334152,
  WL_SHM_FORMAT_BGRA1010102 = 0x30334142, WL_SHM_FORMAT_YUYV = 0x56595559,
  WL_SHM_FORMAT_YVYU = 0x55595659, WL_SHM_FORMAT_UYVY = 0x59565955, WL_SHM_FORMAT_VYUY = 0x59555956, WL_SHM_FORMAT_AYUV = 0x56555941,
  WL_SHM_FORMAT_NV12 = 0x3231564e, WL_SHM_FORMAT_NV21 = 0x3132564e, WL_SHM_FORMAT_NV16 = 0x3631564e, WL_SHM_FORMAT_NV61 = 0x3136564e,
  WL_SHM_FORMAT_YUV410 = 0x39565559, WL_SHM_FORMAT_YVU410 = 0x39555659, WL_SHM_FORMAT_YUV411 = 0x31315559, WL_SHM_FORMAT_YVU411 = 0x31315659,
  WL_SHM_FORMAT_YUV420 = 0x32315559, WL_SHM_FORMAT_YVU420 = 0x32315659, WL_SHM_FORMAT_YUV422 = 0x36315559, WL_SHM_FORMAT_YVU422 = 0x36315659,
  WL_SHM_FORMAT_YUV444 = 0x34325559, WL_SHM_FORMAT_YVU444 = 0x34325659 }

14.13.1 Detailed Description

A singleton global object that provides support for shared memory.

Clients can create wl_shm_pool objects using the create_pool request.

At connection setup time, the wl_shm object emits one or more format events to inform clients about the valid pixel formats that can be used for buffers.

14.13.2 Enumeration Type Documentation

14.13.2.1 wl_shm_error

enum wl_shm_error

wl_shm error values

These errors can be emitted in response to wl_shm requests.

Enumerator

WL_SHM_ERROR_INVALID_FORMAT	buffer format is not known
WL_SHM_ERROR_INVALID_STRIDE	invalid size or stride during pool or buffer creation
WL_SHM_ERROR_INVALID_FD	mmapping the file descriptor failed

14.13.2.2 wl_shm_format

enum wl_shm_format

pixel formats

This describes the memory layout of an individual pixel.

All renderers should support argb8888 and xrgb8888 but any other formats are optional and may not be supported by the particular renderer in use.

The drm format codes match the macros defined in drm_fourcc.h. The formats actually supported by the compositor will be reported by the format event.

Enumerator

WL_SHM_FORMAT_ARGB8888	32-bit ARGB format, [31:0] A:R:G:B 8:8:8:8 little endian
WL_SHM_FORMAT_XRGB8888	32-bit RGB format, [31:0] x:R:G:B 8:8:8:8 little endian
WL_SHM_FORMAT_C8	8-bit color index format, [7:0] C
WL_SHM_FORMAT_RGB332	8-bit RGB format, [7:0] R:G:B 3:3:2
WL_SHM_FORMAT_BGR233	8-bit BGR format, [7:0] B:G:R 2:3:3
WL_SHM_FORMAT_XRGB4444	16-bit xRGB format, [15:0] x:R:G:B 4:4:4:4 little endian
WL_SHM_FORMAT_XBGR4444	16-bit xBGR format, [15:0] x:B:G:R 4:4:4:4 little endian
WL_SHM_FORMAT_RGBX4444	16-bit RGBx format, [15:0] R:G:B:x 4:4:4:4 little endian
WL_SHM_FORMAT_BGRX4444	16-bit BGRx format, [15:0] B:G:R:x 4:4:4:4 little endian
WL_SHM_FORMAT_ARGB4444	16-bit ARGB format, [15:0] A:R:G:B 4:4:4:4 little endian
WL_SHM_FORMAT_ABGR4444	16-bit ABGR format, [15:0] A:B:G:R 4:4:4:4 little endian
WL_SHM_FORMAT_RGBA4444	16-bit RBGA format, [15:0] R:G:B:A 4:4:4:4 little endian
WL_SHM_FORMAT_BGRA4444	16-bit BGRA format, [15:0] B:G:R:A 4:4:4:4 little endian
WL_SHM_FORMAT_XRGB1555	16-bit xRGB format, [15:0] x:R:G:B 1:5:5:5 little endian

Enumerator

WL_SHM_FORMAT_XBGR1555	16-bit xBGR 1555 format, [15:0] x:B:G:R 1:5:5:5 little endian
WL_SHM_FORMAT_RGBX5551	16-bit RGBx 5551 format, [15:0] R:G:B:x 5:5:5:1 little endian
WL_SHM_FORMAT_BGRX5551	16-bit BGRx 5551 format, [15:0] B:G:R:x 5:5:5:1 little endian
WL_SHM_FORMAT_ARGB1555	16-bit ARGB 1555 format, [15:0] A:R:G:B 1:5:5:5 little endian
WL_SHM_FORMAT_ABGR1555	16-bit ABGR 1555 format, [15:0] A:B:G:R 1:5:5:5 little endian
WL_SHM_FORMAT_RGBA5551	16-bit RGBA 5551 format, [15:0] R:G:B:A 5:5:5:1 little endian
WL_SHM_FORMAT_BGRA5551	16-bit BGRA 5551 format, [15:0] B:G:R:A 5:5:5:1 little endian
WL_SHM_FORMAT_RGB565	16-bit RGB 565 format, [15:0] R:G:B 5:6:5 little endian
WL_SHM_FORMAT_BGR565	16-bit BGR 565 format, [15:0] B:G:R 5:6:5 little endian
WL_SHM_FORMAT_RGB888	24-bit RGB format, [23:0] R:G:B little endian
WL_SHM_FORMAT_BGR888	24-bit BGR format, [23:0] B:G:R little endian
WL_SHM_FORMAT_XBGR8888	32-bit xBGR format, [31:0] x:B:G:R 8:8:8:8 little endian
WL_SHM_FORMAT_RGBX8888	32-bit RGBx format, [31:0] R:G:B:x 8:8:8:8 little endian
WL_SHM_FORMAT_BGRX8888	32-bit BGRx format, [31:0] B:G:R:x 8:8:8:8 little endian
WL_SHM_FORMAT_ABGR8888	32-bit ABGR format, [31:0] A:B:G:R 8:8:8:8 little endian
WL_SHM_FORMAT_RGBA8888	32-bit RGBA format, [31:0] R:G:B:A 8:8:8:8 little endian
WL_SHM_FORMAT_BGRA8888	32-bit BGRA format, [31:0] B:G:R:A 8:8:8:8 little endian
WL_SHM_FORMAT_XRGB2101010	32-bit xRGB format, [31:0] x:R:G:B 2:10:10:10 little endian
WL_SHM_FORMAT_XBGR2101010	32-bit xBGR format, [31:0] x:B:G:R 2:10:10:10 little endian
WL_SHM_FORMAT_RGBX1010102	32-bit RGBx format, [31:0] R:G:B:x 10:10:10:2 little endian
WL_SHM_FORMAT_BGRX1010102	32-bit BGRx format, [31:0] B:G:R:x 10:10:10:2 little endian
WL_SHM_FORMAT_ARGB2101010	32-bit ARGB format, [31:0] A:R:G:B 2:10:10:10 little endian
WL_SHM_FORMAT_ABGR2101010	32-bit ABGR format, [31:0] A:B:G:R 2:10:10:10 little endian
WL_SHM_FORMAT_RGBA1010102	32-bit RGBA format, [31:0] R:G:B:A 10:10:10:2 little endian
WL_SHM_FORMAT_BGRA1010102	32-bit BGRA format, [31:0] B:G:R:A 10:10:10:2 little endian
WL_SHM_FORMAT_YUYV	packed YCbCr format, [31:0] Cr0:Y1:Cb0:Y0 8:8:8:8 little endian
WL_SHM_FORMAT_YVYU	packed YCbCr format, [31:0] Cb0:Y1:Cr0:Y0 8:8:8:8 little endian
WL_SHM_FORMAT_UYVY	packed YCbCr format, [31:0] Y1:Cr0:Y0:Cb0 8:8:8:8 little endian
WL_SHM_FORMAT_VYUY	packed YCbCr format, [31:0] Y1:Cb0:Y0:Cr0 8:8:8:8 little endian
WL_SHM_FORMAT_AYUV	packed AYCbCr format, [31:0] A:Y:Cb:Cr 8:8:8:8 little endian
WL_SHM_FORMAT_NV12	2 plane YCbCr Cr:Cb format, 2x2 subsampled Cr:Cb plane
WL_SHM_FORMAT_NV21	2 plane YCbCr Cb:Cr format, 2x2 subsampled Cb:Cr plane
WL_SHM_FORMAT_NV16	2 plane YCbCr Cr:Cb format, 2x1 subsampled Cr:Cb plane
WL_SHM_FORMAT_NV61	2 plane YCbCr Cb:Cr format, 2x1 subsampled Cb:Cr plane
WL_SHM_FORMAT_YUV410	3 plane YCbCr format, 4x4 subsampled Cb (1) and Cr (2) planes
WL_SHM_FORMAT_YVU410	3 plane YCbCr format, 4x4 subsampled Cr (1) and Cb (2) planes
WL_SHM_FORMAT_YUV411	3 plane YCbCr format, 4x1 subsampled Cb (1) and Cr (2) planes
WL_SHM_FORMAT_YVU411	3 plane YCbCr format, 4x1 subsampled Cr (1) and Cb (2) planes
WL_SHM_FORMAT_YUV420	3 plane YCbCr format, 2x2 subsampled Cb (1) and Cr (2) planes
WL_SHM_FORMAT_YVU420	3 plane YCbCr format, 2x2 subsampled Cr (1) and Cb (2) planes
WL_SHM_FORMAT_YUV422	3 plane YCbCr format, 2x1 subsampled Cb (1) and Cr (2) planes
WL_SHM_FORMAT_YVU422	3 plane YCbCr format, 2x1 subsampled Cr (1) and Cb (2) planes
WL_SHM_FORMAT_YUV444	3 plane YCbCr format, non-subsampled Cb (1) and Cr (2) planes
WL_SHM_FORMAT_YVU444	3 plane YCbCr format, non-subsampled Cr (1) and Cb (2) planes

14.14 The wl_buffer interface

Classes

- struct wl_buffer_listener

Macros

- #define **WL_BUFFER_RELEASE_SINCE_VERSION** 1
- #define **WL_BUFFER_DESTROY_SINCE_VERSION** 1

14.14.1 Detailed Description

A buffer provides the content for a wl_surface. Buffers are created through factory interfaces such as wl_drm, wl↩ _shm or similar. It has a width and a height and can be attached to a wl_surface, but the mechanism by which a client provides and updates the contents is defined by the buffer factory interface.

14.15 The wl_data_offer interface

Classes

- struct wl_data_offer_listener

Macros

- #define **WL_DATA_OFFER_OFFER_SINCE_VERSION** 1
- #define **WL_DATA_OFFER_SOURCE_ACTIONS_SINCE_VERSION** 3
- #define **WL_DATA_OFFER_ACTION_SINCE_VERSION** 3
- #define **WL_DATA_OFFER_ACCEPT_SINCE_VERSION** 1
- #define **WL_DATA_OFFER_RECEIVE_SINCE_VERSION** 1
- #define **WL_DATA_OFFER_DESTROY_SINCE_VERSION** 1
- #define **WL_DATA_OFFER_FINISH_SINCE_VERSION** 3
- #define **WL_DATA_OFFER_SET_ACTIONS_SINCE_VERSION** 3

14.15.1 Detailed Description

A wl_data_offer represents a piece of data offered for transfer by another client (the source client). It is used by the copy-and-paste and drag-and-drop mechanisms. The offer describes the different mime types that the data can be converted to and provides the mechanism for transferring the data directly from the source client.

14.16 The wl_data_source interface

Classes

- struct wl_data_source_listener

Macros

- #define **WL_DATA_SOURCE_TARGET_SINCE_VERSION** 1
- #define **WL_DATA_SOURCE_SEND_SINCE_VERSION** 1
- #define **WL_DATA_SOURCE_CANCELLED_SINCE_VERSION** 1
- #define **WL_DATA_SOURCE_DND_DROP_PERFORMED_SINCE_VERSION** 3
- #define **WL_DATA_SOURCE_DND_FINISHED_SINCE_VERSION** 3
- #define **WL_DATA_SOURCE_ACTION_SINCE_VERSION** 3
- #define **WL_DATA_SOURCE_OFFER_SINCE_VERSION** 1
- #define **WL_DATA_SOURCE_DESTROY_SINCE_VERSION** 1
- #define **WL_DATA_SOURCE_SET_ACTIONS_SINCE_VERSION** 3

14.16.1 Detailed Description

The wl_data_source object is the source side of a wl_data_offer. It is created by the source client in a data transfer and provides a way to describe the offered data and a way to respond to requests to transfer the data.

14.17 The wl_data_device interface

Classes

- struct wl_data_device_listener

Macros

- #define **WL_DATA_DEVICE_DATA_OFFER_SINCE_VERSION** 1
- #define **WL_DATA_DEVICE_ENTER_SINCE_VERSION** 1
- #define **WL_DATA_DEVICE_LEAVE_SINCE_VERSION** 1
- #define **WL_DATA_DEVICE_MOTION_SINCE_VERSION** 1
- #define **WL_DATA_DEVICE_DROP_SINCE_VERSION** 1
- #define **WL_DATA_DEVICE_SELECTION_SINCE_VERSION** 1
- #define **WL_DATA_DEVICE_START_DRAG_SINCE_VERSION** 1
- #define **WL_DATA_DEVICE_SET_SELECTION_SINCE_VERSION** 1
- #define **WL_DATA_DEVICE_RELEASE_SINCE_VERSION** 2

14.17.1 Detailed Description

There is one wl_data_device per seat which can be obtained from the global wl_data_device_manager singleton.

A wl_data_device provides access to inter-client data transfer mechanisms such as copy-and-paste and drag-and-drop.

14.18 The wl_data_device_manager interface

Macros

- #define **WL_DATA_DEVICE_MANAGER_CREATE_DATA_SOURCE_SINCE_VERSION** 1
- #define **WL_DATA_DEVICE_MANAGER_GET_DATA_DEVICE_SINCE_VERSION** 1

Enumerations

- enum wl_data_device_manager_dnd_action { WL_DATA_DEVICE_MANAGER_DND_ACTION_NONE = 0, WL_DATA_DEVICE_MANAGER_DND_ACTION_COPY = 1, WL_DATA_DEVICE_MANAGER_DND_ACTION_MOVE = 2, WL_DATA_DEVICE_MANAGER_DND_ACTION_ASK = 4 }

14.18.1 Detailed Description

The wl_data_device_manager is a singleton global object that provides access to inter-client data transfer mechanisms such as copy-and-paste and drag-and-drop. These mechanisms are tied to a wl_seat and this interface lets a client get a wl_data_device corresponding to a wl_seat.

Depending on the version bound, the objects created from the bound wl_data_device_manager object will have different requirements for functioning properly. See wl_data_source.set_actions, wl_data_offer.accept and wl_↩ data_offer.finish for details.

14.18.2 Enumeration Type Documentation

14.18.2.1 wl_data_device_manager_dnd_action

`enum wl_data_device_manager_dnd_action`

drag and drop actions

This is a bitmask of the available/preferred actions in a drag-and-drop operation.

In the compositor, the selected action is a result of matching the actions offered by the source and destination sides. "action" events with a "none" action will be sent to both source and destination if there is no match. All further checks will effectively happen on (source actions ∩ destination actions).

In addition, compositors may also pick different actions in reaction to key modifiers being pressed. One common design that is used in major toolkits (and the behavior recommended for compositors) is:

- If no modifiers are pressed, the first match (in bit order) will be used.
- Pressing Shift selects "move", if enabled in the mask.
- Pressing Control selects "copy", if enabled in the mask.

Behavior beyond that is considered implementation-dependent. Compositors may for example bind other modifiers (like Alt/Meta) or drags initiated with other buttons than BTN_LEFT to specific actions (e.g. "ask").

Enumerator

WL_DATA_DEVICE_MANAGER_DND_ACTION_NONE	no action
WL_DATA_DEVICE_MANAGER_DND_ACTION_COPY	copy action
WL_DATA_DEVICE_MANAGER_DND_ACTION_MOVE	move action
WL_DATA_DEVICE_MANAGER_DND_ACTION_ASK	ask action

14.19 The wl_shell interface

Macros

- #define **WL_SHELL_GET_SHELL_SURFACE_SINCE_VERSION** 1

14.19.1 Detailed Description

This interface is implemented by servers that provide desktop-style user interfaces.

It allows clients to associate a wl_shell_surface with a basic surface.

14.20 The wl_shell_surface interface

Classes

- struct wl_shell_surface_listener

Macros

- #define **WL_SHELL_SURFACE_PING_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_CONFIGURE_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_POPUP_DONE_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_PONG_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_MOVE_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_RESIZE_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_SET_TOPLEVEL_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_SET_TRANSIENT_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_SET_FULLSCREEN_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_SET_POPUP_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_SET_MAXIMIZED_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_SET_TITLE_SINCE_VERSION** 1
- #define **WL_SHELL_SURFACE_SET_CLASS_SINCE_VERSION** 1

Enumerations

- enum wl_shell_surface_resize { WL_SHELL_SURFACE_RESIZE_NONE = 0, WL_SHELL_SURFACE_RESIZE_TOP = 1, WL_SHELL_SURFACE_RESIZE_B = 2, WL_SHELL_SURFACE_RESIZE_LEFT = 4, WL_SHELL_SURFACE_RESIZE_TOP_LEFT = 5, WL_SHELL_SURFACE_RESIZE_BOTTOM_LEFT = 6, WL_SHELL_SURFACE_RESIZE_RIGHT = 8, WL_SHELL_SURFACE_RESIZE_TOP_RIGHT = 9, WL_SHELL_SURFACE_RESIZE_BOTTOM_RIGHT = 10 }
- enum wl_shell_surface_transient { WL_SHELL_SURFACE_TRANSIENT_INACTIVE = 0x1 }
- enum wl_shell_surface_fullscreen_method { WL_SHELL_SURFACE_FULLSCREEN_METHOD_DEFAULT = 0, WL_SHELL_SURFACE_FULLSCREEN_METHOD_SCALE = 1, WL_SHELL_SURFACE_FULLSCREEN_METHOD_DRIV = 2, WL_SHELL_SURFACE_FULLSCREEN_METHOD_FILL = 3 }

14.20.1 Detailed Description

An interface that may be implemented by a wl_surface, for implementations that provide a desktop-style user interface.

It provides requests to treat surfaces like toplevel, fullscreen or popup windows, move, resize or maximize them, associate metadata like title and class, etc.

On the server side the object is automatically destroyed when the related wl_surface is destroyed. On the client side, wl_shell_surface_destroy() must be called before destroying the wl_surface object.

14.20.2 Enumeration Type Documentation

14.20.2.1 wl_shell_surface_fullscreen_method

`enum wl_shell_surface_fullscreen_method`

different method to set the surface fullscreen

Hints to indicate to the compositor how to deal with a conflict between the dimensions of the surface and the dimensions of the output. The compositor is free to ignore this parameter.

Enumerator

WL_SHELL_SURFACE_FULLSCREEN_METHOD↩ _DEFAULT	no preference, apply default policy
WL_SHELL_SURFACE_FULLSCREEN_METHOD↩ _SCALE	scale, preserve the surface's aspect ratio and center on output
WL_SHELL_SURFACE_FULLSCREEN_METHOD↩ _DRIVER	switch output mode to the smallest mode that can fit the surface, add black borders to compensate size mismatch
WL_SHELL_SURFACE_FULLSCREEN_METHOD↩ _FILL	no upscaling, center on output and add black borders to compensate size mismatch

14.20.2.2 wl_shell_surface_resize

`enum wl_shell_surface_resize`

edge values for resizing

These values are used to indicate which edge of a surface is being dragged in a resize operation. The server may use this information to adapt its behavior, e.g. choose an appropriate cursor image.

Enumerator

WL_SHELL_SURFACE_RESIZE_NONE	no edge
WL_SHELL_SURFACE_RESIZE_TOP	top edge
WL_SHELL_SURFACE_RESIZE_BOTTOM	bottom edge
WL_SHELL_SURFACE_RESIZE_LEFT	left edge
WL_SHELL_SURFACE_RESIZE_TOP_LEFT	top and left edges
WL_SHELL_SURFACE_RESIZE_BOTTOM_LEFT	bottom and left edges
WL_SHELL_SURFACE_RESIZE_RIGHT	right edge
WL_SHELL_SURFACE_RESIZE_TOP_RIGHT	top and right edges
WL_SHELL_SURFACE_RESIZE_BOTTOM_RIGHT	bottom and right edges

14.20.2.3 wl_shell_surface_transient

`enum wl_shell_surface_transient`

details of transient behaviour

These flags specify details of the expected behaviour of transient surfaces. Used in the set_transient request.

Enumerator

WL_SHELL_SURFACE_TRANSIENT_INACTIVE	do not set keyboard focus

14.21 The wl_surface interface

Classes

- struct wl_surface_listener

Macros

- #define **WL_SURFACE_ENTER_SINCE_VERSION** 1
- #define **WL_SURFACE_LEAVE_SINCE_VERSION** 1
- #define **WL_SURFACE_DESTROY_SINCE_VERSION** 1
- #define **WL_SURFACE_ATTACH_SINCE_VERSION** 1
- #define **WL_SURFACE_DAMAGE_SINCE_VERSION** 1
- #define **WL_SURFACE_FRAME_SINCE_VERSION** 1
- #define **WL_SURFACE_SET_OPAQUE_REGION_SINCE_VERSION** 1
- #define **WL_SURFACE_SET_INPUT_REGION_SINCE_VERSION** 1
- #define **WL_SURFACE_COMMIT_SINCE_VERSION** 1
- #define **WL_SURFACE_SET_BUFFER_TRANSFORM_SINCE_VERSION** 2
- #define **WL_SURFACE_SET_BUFFER_SCALE_SINCE_VERSION** 3
- #define **WL_SURFACE_DAMAGE_BUFFER_SINCE_VERSION** 4

Enumerations

- enum wl_surface_error { WL_SURFACE_ERROR_INVALID_SCALE = 0, WL_SURFACE_ERROR_INVALID_TRANSFORM = 1 }

14.21.1 Detailed Description

A surface is a rectangular area that is displayed on the screen. It has a location, size and pixel contents.

The size of a surface (and relative positions on it) is described in surface-local coordinates, which may differ from the buffer coordinates of the pixel content, in case a buffer_transform or a buffer_scale is used.

A surface without a "role" is fairly useless: a compositor does not know where, when or how to present it. The role is the purpose of a wl_surface. Examples of roles are a cursor for a pointer (as set by wl_pointer.set_cursor), a drag icon (wl_data_device.start_drag), a sub-surface (wl_subcompositor.get_subsurface), and a window as defined by a shell protocol (e.g. wl_shell.get_shell_surface).

A surface can have only one role at a time. Initially a wl_surface does not have a role. Once a wl_surface is given a role, it is set permanently for the whole lifetime of the wl_surface object. Giving the current role again is allowed, unless explicitly forbidden by the relevant interface specification.

Surface roles are given by requests in other interfaces such as wl_pointer.set_cursor. The request should explicitly mention that this request gives a role to a wl_surface. Often, this request also creates a new protocol object that represents the role and adds additional functionality to wl_surface. When a client wants to destroy a wl_surface, they must destroy this 'role object' before the wl_surface.

Destroying the role object does not remove the role from the wl_surface, but it may stop the wl_surface from "playing the role". For instance, if a wl_subsurface object is destroyed, the wl_surface it was created for will be unmapped and forget its position and z-order. It is allowed to create a wl_subsurface for the same wl_surface again, but it is not allowed to use the wl_surface as a cursor (cursor is a different role than sub-surface, and role switching is not allowed).

14.21.2 Enumeration Type Documentation

14.21.2.1 wl_surface_error

enum wl_surface_error

wl_surface error values

These errors can be emitted in response to wl_surface requests.

Enumerator

WL_SURFACE_ERROR_INVALID_SCALE	buffer scale value is invalid
WL_SURFACE_ERROR_INVALID_TRANSFORM	buffer transform value is invalid

14.22 The wl_seat interface

Classes

- struct wl_seat_listener

Macros

- #define **WL_SEAT_CAPABILITIES_SINCE_VERSION** 1
- #define **WL_SEAT_NAME_SINCE_VERSION** 2
- #define **WL_SEAT_GET_POINTER_SINCE_VERSION** 1
- #define **WL_SEAT_GET_KEYBOARD_SINCE_VERSION** 1
- #define **WL_SEAT_GET_TOUCH_SINCE_VERSION** 1
- #define **WL_SEAT_RELEASE_SINCE_VERSION** 5

Enumerations

- enum wl_seat_capability { WL_SEAT_CAPABILITY_POINTER = 1, WL_SEAT_CAPABILITY_KEYBOARD = 2, WL_SEAT_CAPABILITY_TOUCH = 4 }

14.22.1 Detailed Description

A seat is a group of keyboards, pointer and touch devices. This object is published as a global during start up, or when such a device is hot plugged. A seat typically has a pointer and maintains a keyboard focus and a pointer focus.

14.22.2 Enumeration Type Documentation

14.22.2.1 wl_seat_capability

`enum wl_seat_capability`

seat capability bitmask

This is a bitmask of capabilities this seat has; if a member is set, then it is present on the seat.

Enumerator

WL_SEAT_CAPABILITY_POINTER	the seat has pointer devices
WL_SEAT_CAPABILITY_KEYBOARD	the seat has one or more keyboards
WL_SEAT_CAPABILITY_TOUCH	the seat has touch devices

14.23 The wl_pointer interface

Classes

- struct wl_pointer_listener

Macros

- #define **WL_POINTER_AXIS_SOURCE_WHEEL_TILT_SINCE_VERSION** 6
- #define **WL_POINTER_ENTER_SINCE_VERSION** 1
- #define **WL_POINTER_LEAVE_SINCE_VERSION** 1
- #define **WL_POINTER_MOTION_SINCE_VERSION** 1
- #define **WL_POINTER_BUTTON_SINCE_VERSION** 1
- #define **WL_POINTER_AXIS_SINCE_VERSION** 1
- #define **WL_POINTER_FRAME_SINCE_VERSION** 5
- #define **WL_POINTER_AXIS_SOURCE_SINCE_VERSION** 5
- #define **WL_POINTER_AXIS_STOP_SINCE_VERSION** 5
- #define **WL_POINTER_AXIS_DISCRETE_SINCE_VERSION** 5
- #define **WL_POINTER_SET_CURSOR_SINCE_VERSION** 1
- #define **WL_POINTER_RELEASE_SINCE_VERSION** 3

Enumerations

- enum wl_pointer_button_state { WL_POINTER_BUTTON_STATE_RELEASED = 0, WL_POINTER_BUTTON_STATE_PRESS = 1 }
- enum wl_pointer_axis { WL_POINTER_AXIS_VERTICAL_SCROLL = 0, WL_POINTER_AXIS_HORIZONTAL_SCROLL = 1 }
- enum wl_pointer_axis_source { WL_POINTER_AXIS_SOURCE_WHEEL = 0, WL_POINTER_AXIS_SOURCE_FINGER = 1, WL_POINTER_AXIS_SOURCE_CONTINUOUS = 2, WL_POINTER_AXIS_SOURCE_WHEEL_TILT = 3 }

14.23.1 Detailed Description

The wl_pointer interface represents one or more input devices, such as mice, which control the pointer location and pointer_focus of a seat.

The wl_pointer interface generates motion, enter and leave events for the surfaces that the pointer is located over, and button and axis events for button presses, button releases and scrolling.

14.23.2 Enumeration Type Documentation

14.23.2.1 wl_pointer_axis

```
enum wl_pointer_axis
```

axis types

Describes the axis types of scroll events.

Enumerator

WL_POINTER_AXIS_VERTICAL_SCROLL	vertical axis
WL_POINTER_AXIS_HORIZONTAL_SCROLL	horizontal axis

14.23.2.2 wl_pointer_axis_source

`enum wl_pointer_axis_source`

axis source types

Describes the source types for axis events. This indicates to the client how an axis event was physically generated; a client may adjust the user interface accordingly. For example, scroll events from a "finger" source may be in a smooth coordinate space with kinetic scrolling whereas a "wheel" source may be in discrete steps of a number of lines.

The "continuous" axis source is a device generating events in a continuous coordinate space, but using something other than a finger. One example for this source is button-based scrolling where the vertical motion of a device is converted to scroll events while a button is held down.

The "wheel tilt" axis source indicates that the actual device is a wheel but the scroll event is not caused by a rotation but a (usually sideways) tilt of the wheel.

Enumerator

WL_POINTER_AXIS_SOURCE_WHEEL	a physical wheel rotation
WL_POINTER_AXIS_SOURCE_FINGER	finger on a touch surface
WL_POINTER_AXIS_SOURCE_CONTINUOUS	continuous coordinate space
WL_POINTER_AXIS_SOURCE_WHEEL_TILT	a physical wheel tilt **Since** 6

14.23.2.3 wl_pointer_button_state

`enum wl_pointer_button_state`

physical button state

Describes the physical state of a button that produced the button event.

Enumerator

WL_POINTER_BUTTON_STATE_RELEASED	the button is not pressed
WL_POINTER_BUTTON_STATE_PRESSED	the button is pressed

14.24 The wl_keyboard interface

Classes

- struct wl_keyboard_listener

Macros

- #define **WL_KEYBOARD_KEYMAP_SINCE_VERSION** 1
- #define **WL_KEYBOARD_ENTER_SINCE_VERSION** 1
- #define **WL_KEYBOARD_LEAVE_SINCE_VERSION** 1
- #define **WL_KEYBOARD_KEY_SINCE_VERSION** 1
- #define **WL_KEYBOARD_MODIFIERS_SINCE_VERSION** 1
- #define **WL_KEYBOARD_REPEAT_INFO_SINCE_VERSION** 4
- #define **WL_KEYBOARD_RELEASE_SINCE_VERSION** 3

Enumerations

- enum wl_keyboard_keymap_format { WL_KEYBOARD_KEYMAP_FORMAT_NO_KEYMAP = 0, WL_KEYBOARD_KEYMAP_F = 1 }
- enum wl_keyboard_key_state { WL_KEYBOARD_KEY_STATE_RELEASED = 0, WL_KEYBOARD_KEY_STATE_PRESSED = 1 }

14.24.1 Detailed Description

The wl_keyboard interface represents one or more keyboards associated with a seat.

14.24.2 Enumeration Type Documentation

14.24.2.1 wl_keyboard_key_state

`enum wl_keyboard_key_state`

physical key state

Describes the physical state of a key that produced the key event.

Enumerator

WL_KEYBOARD_KEY_STATE_RELEASED	key is not pressed
WL_KEYBOARD_KEY_STATE_PRESSED	key is pressed

14.24.2.2 wl_keyboard_keymap_format

```
enum wl_keyboard_keymap_format
```

keyboard mapping format

This specifies the format of the keymap provided to the client with the wl_keyboard.keymap event.

Enumerator

WL_KEYBOARD_KEYMAP_FORMAT_NO_KEYMAP	no keymap; client must understand how to interpret the raw keycode
WL_KEYBOARD_KEYMAP_FORMAT_XKB_V1	libxkbcommon compatible; to determine the xkb keycode, clients must add 8 to the key event keycode

14.25 The wl_touch interface

Classes

- struct wl_touch_listener

Macros

- #define **WL_TOUCH_DOWN_SINCE_VERSION** 1
- #define **WL_TOUCH_UP_SINCE_VERSION** 1
- #define **WL_TOUCH_MOTION_SINCE_VERSION** 1
- #define **WL_TOUCH_FRAME_SINCE_VERSION** 1
- #define **WL_TOUCH_CANCEL_SINCE_VERSION** 1
- #define **WL_TOUCH_SHAPE_SINCE_VERSION** 6
- #define **WL_TOUCH_ORIENTATION_SINCE_VERSION** 6
- #define **WL_TOUCH_RELEASE_SINCE_VERSION** 3

14.25.1 Detailed Description

The wl_touch interface represents a touchscreen associated with a seat.

Touch interactions can consist of one or more contacts. For each contact, a series of events is generated, starting with a down event, followed by zero or more motion events, and ending with an up event. Events relating to the same contact point can be identified by the ID of the sequence.

14.26 The wl_output interface

Classes

- struct wl_output_listener

Macros

- #define **WL_OUTPUT_GEOMETRY_SINCE_VERSION** 1
- #define **WL_OUTPUT_MODE_SINCE_VERSION** 1
- #define **WL_OUTPUT_DONE_SINCE_VERSION** 2
- #define **WL_OUTPUT_SCALE_SINCE_VERSION** 2
- #define **WL_OUTPUT_RELEASE_SINCE_VERSION** 3

Enumerations

- enum wl_output_subpixel { WL_OUTPUT_SUBPIXEL_UNKNOWN = 0, WL_OUTPUT_SUBPIXEL_NONE = 1, WL_OUTPUT_SUBPIXEL_HORIZONTAL_ = 2, WL_OUTPUT_SUBPIXEL_HORIZONTAL_BGR = 3, WL_OUTPUT_SUBPIXEL_VERTICAL_RGB = 4, WL_OUTPUT_SUBPIXEL_VERTICAL_BGR = 5 }
- enum wl_output_transform { WL_OUTPUT_TRANSFORM_NORMAL = 0, WL_OUTPUT_TRANSFORM_90 = 1, WL_OUTPUT_TRANSFORM_180 = 2, WL_OUTPUT_TRANSFORM_270 = 3, WL_OUTPUT_TRANSFORM_FLIPPED = 4, WL_OUTPUT_TRANSFORM_FLIPPED_90 = 5, WL_OUTPUT_TRANSFORM_F = 6, WL_OUTPUT_TRANSFORM_FLIPPED_270 = 7 }
- enum wl_output_mode { WL_OUTPUT_MODE_CURRENT = 0x1, WL_OUTPUT_MODE_PREFERRED = 0x2 }

14.26.1 Detailed Description

An output describes part of the compositor geometry. The compositor works in the 'compositor coordinate system' and an output corresponds to a rectangular area in that space that is actually visible. This typically corresponds to a monitor that displays part of the compositor space. This object is published as global during start up, or when a monitor is hotplugged.

14.26.2 Enumeration Type Documentation

14.26.2.1 wl_output_mode

```
enum wl_output_mode
```

mode information

These flags describe properties of an output mode. They are used in the flags bitfield of the mode event.

Enumerator

WL_OUTPUT_MODE_CURRENT	indicates this is the current mode
WL_OUTPUT_MODE_PREFERRED	indicates this is the preferred mode

14.26.2.2 wl_output_subpixel

`enum wl_output_subpixel`

subpixel geometry information

This enumeration describes how the physical pixels on an output are laid out.

Enumerator

WL_OUTPUT_SUBPIXEL_UNKNOWN	unknown geometry
WL_OUTPUT_SUBPIXEL_NONE	no geometry
WL_OUTPUT_SUBPIXEL_HORIZONTAL_RGB	horizontal RGB
WL_OUTPUT_SUBPIXEL_HORIZONTAL_BGR	horizontal BGR
WL_OUTPUT_SUBPIXEL_VERTICAL_RGB	vertical RGB
WL_OUTPUT_SUBPIXEL_VERTICAL_BGR	vertical BGR

14.26.2.3 wl_output_transform

`enum wl_output_transform`

transform from framebuffer to output

This describes the transform that a compositor will apply to a surface to compensate for the rotation or mirroring of an output device.

The flipped values correspond to an initial flip around a vertical axis followed by rotation.

The purpose is mainly to allow clients to render accordingly and tell the compositor, so that for fullscreen surfaces, the compositor will still be able to scan out directly from client surfaces.

Enumerator

WL_OUTPUT_TRANSFORM_NORMAL	no transform
WL_OUTPUT_TRANSFORM_90	90 degrees counter-clockwise
WL_OUTPUT_TRANSFORM_180	180 degrees counter-clockwise
WL_OUTPUT_TRANSFORM_270	270 degrees counter-clockwise
WL_OUTPUT_TRANSFORM_FLIPPED	180 degree flip around a vertical axis
WL_OUTPUT_TRANSFORM_FLIPPED_90	flip and rotate 90 degrees counter-clockwise
WL_OUTPUT_TRANSFORM_FLIPPED_180	flip and rotate 180 degrees counter-clockwise
WL_OUTPUT_TRANSFORM_FLIPPED_270	flip and rotate 270 degrees counter-clockwise

14.27 The wl_region interface

Macros

- #define **WL_REGION_DESTROY_SINCE_VERSION** 1
- #define **WL_REGION_ADD_SINCE_VERSION** 1
- #define **WL_REGION_SUBTRACT_SINCE_VERSION** 1

14.27.1 Detailed Description

A region object describes an area.

Region objects are used to describe the opaque and input regions of a surface.

14.28 The wl_subcompositor interface

Macros

- #define **WL_SUBCOMPOSITOR_DESTROY_SINCE_VERSION** 1
- #define **WL_SUBCOMPOSITOR_GET_SUBSURFACE_SINCE_VERSION** 1

14.28.1 Detailed Description

The global interface exposing sub-surface compositing capabilities. A wl_surface, that has sub-surfaces associated, is called the parent surface. Sub-surfaces can be arbitrarily nested and create a tree of sub-surfaces.

The root surface in a tree of sub-surfaces is the main surface. The main surface cannot be a sub-surface, because sub-surfaces must always have a parent.

A main surface with its sub-surfaces forms a (compound) window. For window management purposes, this set of wl_surface objects is to be considered as a single window, and it should also behave as such.

The aim of sub-surfaces is to offload some of the compositing work within a window from clients to the compositor. A prime example is a video player with decorations and video in separate wl_surface objects. This should allow the compositor to pass YUV video buffer processing to dedicated overlay hardware when possible.

14.29 The wl_subsurface interface

Macros

- #define **WL_SUBSURFACE_DESTROY_SINCE_VERSION** 1
- #define **WL_SUBSURFACE_SET_POSITION_SINCE_VERSION** 1
- #define **WL_SUBSURFACE_PLACE_ABOVE_SINCE_VERSION** 1
- #define **WL_SUBSURFACE_PLACE_BELOW_SINCE_VERSION** 1
- #define **WL_SUBSURFACE_SET_SYNC_SINCE_VERSION** 1
- #define **WL_SUBSURFACE_SET_DESYNC_SINCE_VERSION** 1

14.29.1 Detailed Description

An additional interface to a wl_surface object, which has been made a sub-surface. A sub-surface has one parent surface. A sub-surface's size and position are not limited to that of the parent. Particularly, a sub-surface is not automatically clipped to its parent's area.

A sub-surface becomes mapped, when a non-NULL wl_buffer is applied and the parent surface is mapped. The order of which one happens first is irrelevant. A sub-surface is hidden if the parent becomes hidden, or if a NULL wl_buffer is applied. These rules apply recursively through the tree of surfaces.

The behaviour of a wl_surface.commit request on a sub-surface depends on the sub-surface's mode. The possible modes are synchronized and desynchronized, see methods wl_subsurface.set_sync and wl_subsurface.set_↩ desync. Synchronized mode caches the wl_surface state to be applied when the parent's state gets applied, and desynchronized mode applies the pending wl_surface state directly. A sub-surface is initially in the synchronized mode.

Sub-surfaces have also other kind of state, which is managed by wl_subsurface requests, as opposed to wl_surface requests. This state includes the sub-surface position relative to the parent surface (wl_subsurface.set_position), and the stacking order of the parent and its sub-surfaces (wl_subsurface.place_above and .place_below). This state is applied when the parent surface's wl_surface state is applied, regardless of the sub-surface's mode. As the exception, set_sync and set_desync are effective immediately.

The main surface can be thought to be always in desynchronized mode, since it does not have a parent in the sub-surfaces sense.

Even if a sub-surface is in desynchronized mode, it will behave as in synchronized mode, if its parent surface behaves as in synchronized mode. This rule is applied recursively throughout the tree of surfaces. This means, that one can set a sub-surface into synchronized mode, and then assume that all its child and grand-child sub-surfaces are synchronized, too, without explicitly setting them.

If the wl_surface associated with the wl_subsurface is destroyed, the wl_subsurface object becomes inert. Note, that destroying either object takes effect immediately. If you need to synchronize the removal of a sub-surface to the parent surface update, unmap the sub-surface first by attaching a NULL wl_buffer, update parent, and then destroy the sub-surface.

If the parent wl_surface object is destroyed, the sub-surface is unmapped.

14.30 The zxdg_decoration_manager_v1 interface

Macros

- #define **ZXDG_DECORATION_MANAGER_V1_DESTROY_SINCE_VERSION** 1
- #define **ZXDG_DECORATION_MANAGER_V1_GET_TOPLEVEL_DECORATION_SINCE_VERSION** 1

14.30.1 Detailed Description

This interface allows a compositor to announce support for server-side decorations.

A window decoration is a set of window controls as deemed appropriate by the party managing them, such as user interface components used to move, resize and change a window's state.

A client can use this protocol to request being decorated by a supporting compositor.

If compositor and client do not negotiate the use of a server-side decoration using this protocol, clients continue to self-decorate as they see fit.

Warning! The protocol described in this file is experimental and backward incompatible changes may be made. Backward compatible changes may be added together with the corresponding interface version bump. Backward incompatible changes are done by bumping the version number in the protocol and interface names and resetting the interface version. Once the protocol is to be declared stable, the 'z' prefix and the version number in the protocol and interface names are removed and the interface version number is reset.

14.31 The zxdg_toplevel_decoration_v1 interface

Classes

- struct zxdg_toplevel_decoration_v1_listener

Macros

- #define **ZXDG_TOPLEVEL_DECORATION_V1_CONFIGURE_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_DECORATION_V1_DESTROY_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_DECORATION_V1_SET_MODE_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_DECORATION_V1_UNSET_MODE_SINCE_VERSION** 1

Enumerations

- enum zxdg_toplevel_decoration_v1_mode { ZXDG_TOPLEVEL_DECORATION_V1_MODE_CLIENT_SIDE = 1, ZXDG_TOPLEVEL_DECORATION_V1_MODE_SERVER_SIDE = 2 }

14.31.1 Detailed Description

The decoration object allows the compositor to toggle server-side window decorations for a toplevel surface. The client can request to switch to another mode.

The xdg_toplevel_decoration object must be destroyed before its xdg_toplevel.

14.31.2 Enumeration Type Documentation

14.31.2.1 zxdg_toplevel_decoration_v1_mode

```
enum zxdg_toplevel_decoration_v1_mode
```

window decoration modes

These values describe window decoration modes.

Enumerator

ZXDG_TOPLEVEL_DECORATION_V1_MODE_CLIENT_SIDE	no server-side window decoration
ZXDG_TOPLEVEL_DECORATION_V1_MODE_SERVER_SIDE	server-side window decoration

14.32 The xdg_wm_base interface

Classes

- struct xdg_wm_base_listener

Macros

- #define **XDG_WM_BASE_PING_SINCE_VERSION** 1
- #define **XDG_WM_BASE_DESTROY_SINCE_VERSION** 1
- #define **XDG_WM_BASE_CREATE_POSITIONER_SINCE_VERSION** 1
- #define **XDG_WM_BASE_GET_XDG_SURFACE_SINCE_VERSION** 1
- #define **XDG_WM_BASE_PONG_SINCE_VERSION** 1

14.32.1 Detailed Description

The xdg_wm_base interface is exposed as a global object enabling clients to turn their wl_surfaces into windows in a desktop environment. It defines the basic functionality needed for clients and the compositor to create windows that can be dragged, resized, maximized, etc, as well as creating transient windows such as popup menus.

14.33 The xdg_positioner interface

Macros

- #define **XDG_POSITIONER_DESTROY_SINCE_VERSION** 1
- #define **XDG_POSITIONER_SET_SIZE_SINCE_VERSION** 1
- #define **XDG_POSITIONER_SET_ANCHOR_RECT_SINCE_VERSION** 1
- #define **XDG_POSITIONER_SET_ANCHOR_SINCE_VERSION** 1
- #define **XDG_POSITIONER_SET_GRAVITY_SINCE_VERSION** 1
- #define **XDG_POSITIONER_SET_CONSTRAINT_ADJUSTMENT_SINCE_VERSION** 1
- #define **XDG_POSITIONER_SET_OFFSET_SINCE_VERSION** 1

Enumerations

- enum xdg_positioner_constraint_adjustment {
  XDG_POSITIONER_CONSTRAINT_ADJUSTMENT_NONE = 0, **XDG_POSITIONER_CONSTRAINT_A↩
  DJUSTMENT_SLIDE_X** = 1, **XDG_POSITIONER_CONSTRAINT_ADJUSTMENT_SLIDE_Y** = 2, **XDG_P↩
  OSITIONER_CONSTRAINT_ADJUSTMENT_FLIP_X** = 4,
  XDG_POSITIONER_CONSTRAINT_ADJUSTMENT_FLIP_Y = 8, **XDG_POSITIONER_CONSTRAINT_A↩
  DJUSTMENT_RESIZE_X** = 16, **XDG_POSITIONER_CONSTRAINT_ADJUSTMENT_RESIZE_Y** = 32 }

14.33.1 Detailed Description

The xdg_positioner provides a collection of rules for the placement of a child surface relative to a parent surface. Rules can be defined to ensure the child surface remains within the visible area's borders, and to specify how the child surface changes its position, such as sliding along an axis, or flipping around a rectangle. These positioner-created rules are constrained by the requirement that a child surface must intersect with or be at least partially adjacent to its parent surface.

See the various requests for details about possible rules.

At the time of the request, the compositor makes a copy of the rules specified by the xdg_positioner. Thus, after the request is complete the xdg_positioner object can be destroyed or reused; further changes to the object will have no effect on previous usages.

For an xdg_positioner object to be considered complete, it must have a non-zero size set by set_size, and a non-zero anchor rectangle set by set_anchor_rect. Passing an incomplete xdg_positioner object when positioning a surface raises an error.

14.33.2 Enumeration Type Documentation

14.33.2.1 xdg_positioner_constraint_adjustment

enum xdg_positioner_constraint_adjustment

vertically resize the surface

Resize the surface vertically so that it is completely unconstrained.

14.34 The xdg_surface interface

Classes

- struct xdg_surface_listener

Macros

- #define **XDG_SURFACE_CONFIGURE_SINCE_VERSION** 1
- #define **XDG_SURFACE_DESTROY_SINCE_VERSION** 1
- #define **XDG_SURFACE_GET_TOPLEVEL_SINCE_VERSION** 1
- #define **XDG_SURFACE_GET_POPUP_SINCE_VERSION** 1
- #define **XDG_SURFACE_SET_WINDOW_GEOMETRY_SINCE_VERSION** 1
- #define **XDG_SURFACE_ACK_CONFIGURE_SINCE_VERSION** 1

14.34.1 Detailed Description

An interface that may be implemented by a wl_surface, for implementations that provide a desktop-style user interface.

It provides a base set of functionality required to construct user interface elements requiring management by the compositor, such as toplevel windows, menus, etc. The types of functionality are split into xdg_surface roles.

Creating an xdg_surface does not set the role for a wl_surface. In order to map an xdg_surface, the client must create a role-specific object using, e.g., get_toplevel, get_popup. The wl_surface for any given xdg_surface can have at most one role, and may not be assigned any role not based on xdg_surface.

A role must be assigned before any other requests are made to the xdg_surface object.

The client must call wl_surface.commit on the corresponding wl_surface for the xdg_surface state to take effect.

Creating an xdg_surface from a wl_surface which has a buffer attached or committed is a client error, and any attempts by a client to attach or manipulate a buffer prior to the first xdg_surface.configure call must also be treated as errors.

Mapping an xdg_surface-based role surface is defined as making it possible for the surface to be shown by the compositor. Note that a mapped surface is not guaranteed to be visible once it is mapped.

For an xdg_surface to be mapped by the compositor, the following conditions must be met: (1) the client has assigned an xdg_surface-based role to the surface (2) the client has set and committed the xdg_surface state and the role-dependent state to the surface (3) the client has committed a buffer to the surface

A newly-unmapped surface is considered to have met condition (1) out of the 3 required conditions for mapping a surface if its role surface has not been destroyed.

14.35 The xdg_toplevel interface

Classes

- struct xdg_toplevel_listener

Macros

- #define **XDG_TOPLEVEL_CONFIGURE_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_CLOSE_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_DESTROY_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_SET_PARENT_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_SET_TITLE_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_SET_APP_ID_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_SHOW_WINDOW_MENU_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_MOVE_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_RESIZE_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_SET_MAX_SIZE_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_SET_MIN_SIZE_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_SET_MAXIMIZED_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_UNSET_MAXIMIZED_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_SET_FULLSCREEN_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_UNSET_FULLSCREEN_SINCE_VERSION** 1
- #define **XDG_TOPLEVEL_SET_MINIMIZED_SINCE_VERSION** 1

Enumerations

- enum xdg_toplevel_resize_edge { **XDG_TOPLEVEL_RESIZE_EDGE_NONE** = 0, **XDG_TOPLEVEL_RESIZE_EDGE_TOP** = 1, **XDG_TOPLEVEL_RESIZE_EDGE_BOTTOM** = 2, **XDG_TOPLEVEL_RESIZE_EDGE_LEFT** = 4, **XDG_TOPLEVEL_RESIZE_EDGE_TOP_LEFT** = 5, **XDG_TOPLEVEL_RESIZE_EDGE_BOTTOM_LEFT** = 6, **XDG_TOPLEVEL_RESIZE_EDGE_RIGHT** = 8, **XDG_TOPLEVEL_RESIZE_EDGE_TOP_RIGHT** = 9, **XDG_TOPLEVEL_RESIZE_EDGE_BOTTOM_RIGHT** = 10 }
- enum xdg_toplevel_state { XDG_TOPLEVEL_STATE_MAXIMIZED = 1, XDG_TOPLEVEL_STATE_FULLSCREEN = 2, XDG_TOPLEVEL_STATE_RESIZING = 3, XDG_TOPLEVEL_STATE_ACTIVATED = 4 }

14.35.1 Detailed Description

This interface defines an xdg_surface role which allows a surface to, among other things, set window-like properties such as maximize, fullscreen, and minimize, set application-specific metadata like title and id, and well as trigger user interactive operations such as interactive resize and move.

Unmapping an xdg_toplevel means that the surface cannot be shown by the compositor until it is explicitly mapped again. All active operations (e.g., move, resize) are canceled and all attributes (e.g. title, state, stacking, ...) are discarded for an xdg_toplevel surface when it is unmapped.

Attaching a null buffer to a toplevel unmaps the surface.

14.35.2 Enumeration Type Documentation

14.35.2.1 xdg_toplevel_resize_edge

enum xdg_toplevel_resize_edge

edge values for resizing

These values are used to indicate which edge of a surface is being dragged in a resize operation.

14.35.2.2 xdg_toplevel_state

enum xdg_toplevel_state

the surface is now activated

Client window decorations should be painted as if the window is active. Do not assume this means that the window actually has keyboard or pointer focus.

Enumerator

XDG_TOPLEVEL_STATE_MAXIMIZED	the surface is maximized
XDG_TOPLEVEL_STATE_FULLSCREEN	the surface is fullscreen
XDG_TOPLEVEL_STATE_RESIZING	the surface is being resized
XDG_TOPLEVEL_STATE_ACTIVATED	the surface is now activated

14.36 The xdg_popup interface

Classes

- struct xdg_popup_listener

Macros

- #define **XDG_POPUP_CONFIGURE_SINCE_VERSION** 1
- #define **XDG_POPUP_POPUP_DONE_SINCE_VERSION** 1
- #define **XDG_POPUP_DESTROY_SINCE_VERSION** 1
- #define **XDG_POPUP_GRAB_SINCE_VERSION** 1

14.36.1 Detailed Description

A popup surface is a short-lived, temporary surface. It can be used to implement for example menus, popovers, tooltips and other similar user interface concepts.

A popup can be made to take an explicit grab. See xdg_popup.grab for details.

When the popup is dismissed, a popup_done event will be sent out, and at the same time the surface will be unmapped. See the xdg_popup.popup_done event for details.

Explicitly destroying the xdg_popup object will also dismiss the popup and unmap the surface. Clients that want to dismiss the popup when another surface of their own is clicked should dismiss the popup using the destroy request.

The parent surface must have either the xdg_toplevel or xdg_popup surface role.

A newly created xdg_popup will be stacked on top of all previously created xdg_popup surfaces associated with the same xdg_toplevel.

The parent of an xdg_popup must be mapped (see the xdg_surface description) before the xdg_popup itself.

The x and y arguments passed when creating the popup object specify where the top left of the popup should be placed, relative to the local surface coordinates of the parent surface. See xdg_surface.get_popup. An xdg_popup must intersect with or be at least partially adjacent to its parent surface.

The client must call wl_surface.commit on the corresponding wl_surface for the xdg_popup state to take effect.

14.37 The zxdg_shell_v6 interface

Classes

- struct zxdg_shell_v6_listener

Macros

- #define **ZXDG_SHELL_V6_PING_SINCE_VERSION** 1
- #define **ZXDG_SHELL_V6_DESTROY_SINCE_VERSION** 1
- #define **ZXDG_SHELL_V6_CREATE_POSITIONER_SINCE_VERSION** 1
- #define **ZXDG_SHELL_V6_GET_XDG_SURFACE_SINCE_VERSION** 1
- #define **ZXDG_SHELL_V6_PONG_SINCE_VERSION** 1

14.37.1 Detailed Description

xdg_shell allows clients to turn a wl_surface into a "real window" which can be dragged, resized, stacked, and moved around by the user. Everything about this interface is suited towards traditional desktop environments.

14.38 The zxdg_positioner_v6 interface

Macros

- #define **ZXDG_POSITIONER_V6_DESTROY_SINCE_VERSION** 1
- #define **ZXDG_POSITIONER_V6_SET_SIZE_SINCE_VERSION** 1
- #define **ZXDG_POSITIONER_V6_SET_ANCHOR_RECT_SINCE_VERSION** 1
- #define **ZXDG_POSITIONER_V6_SET_ANCHOR_SINCE_VERSION** 1
- #define **ZXDG_POSITIONER_V6_SET_GRAVITY_SINCE_VERSION** 1
- #define **ZXDG_POSITIONER_V6_SET_CONSTRAINT_ADJUSTMENT_SINCE_VERSION** 1
- #define **ZXDG_POSITIONER_V6_SET_OFFSET_SINCE_VERSION** 1

Enumerations

- enum zxdg_positioner_v6_constraint_adjustment {
  ZXDG_POSITIONER_V6_CONSTRAINT_ADJUSTMENT_NONE = 0, **ZXDG_POSITIONER_V6_CONSTRAINT_ADJUSTMENT_SLIDE_X** = 1, **ZXDG_POSITIONER_V6_CONSTRAINT_ADJUSTMENT_SLIDE_Y** = 2, **ZXDG_POSITIONER_V6_CONSTRAINT_ADJUSTMENT_FLIP_X** = 4, **ZXDG_POSITIONER_V6_CONSTRAINT_ADJUSTMENT_FLIP_Y** = 8, **ZXDG_POSITIONER_V6_CONSTRAINT_ADJUSTMENT_RESIZE_X** = 16, **ZXDG_POSITIONER_V6_CONSTRAINT_ADJUSTMENT_RESIZE_Y** = 32 }

14.38.1 Detailed Description

The xdg_positioner provides a collection of rules for the placement of a child surface relative to a parent surface. Rules can be defined to ensure the child surface remains within the visible area's borders, and to specify how the child surface changes its position, such as sliding along an axis, or flipping around a rectangle. These positioner-created rules are constrained by the requirement that a child surface must intersect with or be at least partially adjacent to its parent surface.

See the various requests for details about possible rules.

At the time of the request, the compositor makes a copy of the rules specified by the xdg_positioner. Thus, after the request is complete the xdg_positioner object can be destroyed or reused; further changes to the object will have no effect on previous usages.

For an xdg_positioner object to be considered complete, it must have a non-zero size set by set_size, and a non-zero anchor rectangle set by set_anchor_rect. Passing an incomplete xdg_positioner object when positioning a surface raises an error.

14.38.2 Enumeration Type Documentation

14.38.2.1 zxdg_positioner_v6_constraint_adjustment

enum zxdg_positioner_v6_constraint_adjustment

vertically resize the surface

Resize the surface vertically so that it is completely unconstrained.

14.39 The zxdg_surface_v6 interface

Classes

- struct zxdg_surface_v6_listener

Macros

- #define **ZXDG_SURFACE_V6_CONFIGURE_SINCE_VERSION** 1
- #define **ZXDG_SURFACE_V6_DESTROY_SINCE_VERSION** 1
- #define **ZXDG_SURFACE_V6_GET_TOPLEVEL_SINCE_VERSION** 1
- #define **ZXDG_SURFACE_V6_GET_POPUP_SINCE_VERSION** 1
- #define **ZXDG_SURFACE_V6_SET_WINDOW_GEOMETRY_SINCE_VERSION** 1
- #define **ZXDG_SURFACE_V6_ACK_CONFIGURE_SINCE_VERSION** 1

14.39.1 Detailed Description

An interface that may be implemented by a wl_surface, for implementations that provide a desktop-style user interface.

It provides a base set of functionality required to construct user interface elements requiring management by the compositor, such as toplevel windows, menus, etc. The types of functionality are split into xdg_surface roles.

Creating an xdg_surface does not set the role for a wl_surface. In order to map an xdg_surface, the client must create a role-specific object using, e.g., get_toplevel, get_popup. The wl_surface for any given xdg_surface can have at most one role, and may not be assigned any role not based on xdg_surface.

A role must be assigned before any other requests are made to the xdg_surface object.

The client must call wl_surface.commit on the corresponding wl_surface for the xdg_surface state to take effect.

Creating an xdg_surface from a wl_surface which has a buffer attached or committed is a client error, and any attempts by a client to attach or manipulate a buffer prior to the first xdg_surface.configure call must also be treated as errors.

For a surface to be mapped by the compositor, the following conditions must be met: (1) the client has assigned a xdg_surface based role to the surface, (2) the client has set and committed the xdg_surface state and the role dependent state to the surface and (3) the client has committed a buffer to the surface.

14.40 The zxdg_toplevel_v6 interface

Classes

- struct zxdg_toplevel_v6_listener

Macros

- #define **ZXDG_TOPLEVEL_V6_CONFIGURE_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_CLOSE_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_DESTROY_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_SET_PARENT_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_SET_TITLE_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_SET_APP_ID_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_SHOW_WINDOW_MENU_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_MOVE_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_RESIZE_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_SET_MAX_SIZE_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_SET_MIN_SIZE_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_SET_MAXIMIZED_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_UNSET_MAXIMIZED_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_SET_FULLSCREEN_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_UNSET_FULLSCREEN_SINCE_VERSION** 1
- #define **ZXDG_TOPLEVEL_V6_SET_MINIMIZED_SINCE_VERSION** 1

Enumerations

- enum zxdg_toplevel_v6_resize_edge {
  ZXDG_TOPLEVEL_V6_RESIZE_EDGE_NONE = 0, **ZXDG_TOPLEVEL_V6_RESIZE_EDGE_TOP** = 1, **ZXDG_TOPLEVEL_V6_RESIZE_EDGE_BOTTOM** = 2, **ZXDG_TOPLEVEL_V6_RESIZE_EDGE_LEFT** = 4,
  ZXDG_TOPLEVEL_V6_RESIZE_EDGE_TOP_LEFT = 5, **ZXDG_TOPLEVEL_V6_RESIZE_EDGE_BOTTOM_LEFT** = 6, **ZXDG_TOPLEVEL_V6_RESIZE_EDGE_RIGHT** = 8, **ZXDG_TOPLEVEL_V6_RESIZE_EDGE_TOP_RIGHT** = 9,
  ZXDG_TOPLEVEL_V6_RESIZE_EDGE_BOTTOM_RIGHT = 10 }
- enum zxdg_toplevel_v6_state { ZXDG_TOPLEVEL_V6_STATE_MAXIMIZED = 1, ZXDG_TOPLEVEL_V6_STATE_FULLSCRE = 2, ZXDG_TOPLEVEL_V6_STATE_RESIZING = 3, ZXDG_TOPLEVEL_V6_STATE_ACTIVATED = 4 }

14.40.1 Detailed Description

This interface defines an xdg_surface role which allows a surface to, among other things, set window-like properties such as maximize, fullscreen, and minimize, set application-specific metadata like title and id, and well as trigger user interactive operations such as interactive resize and move.

14.40.2 Enumeration Type Documentation

14.40.2.1 zxdg_toplevel_v6_resize_edge

enum zxdg_toplevel_v6_resize_edge

edge values for resizing

These values are used to indicate which edge of a surface is being dragged in a resize operation.

14.40.2.2 zxdg_toplevel_v6_state

enum zxdg_toplevel_v6_state

the surface is now activated

Client window decorations should be painted as if the window is active. Do not assume this means that the window actually has keyboard or pointer focus.

Enumerator

ZXDG_TOPLEVEL_V6_STATE_MAXIMIZED	the surface is maximized
ZXDG_TOPLEVEL_V6_STATE_FULLSCREEN	the surface is fullscreen
ZXDG_TOPLEVEL_V6_STATE_RESIZING	the surface is being resized
ZXDG_TOPLEVEL_V6_STATE_ACTIVATED	the surface is now activated

14.41 The zxdg_popup_v6 interface

Classes

- struct zxdg_popup_v6_listener

Macros

- #define **ZXDG_POPUP_V6_CONFIGURE_SINCE_VERSION** 1
- #define **ZXDG_POPUP_V6_POPUP_DONE_SINCE_VERSION** 1
- #define **ZXDG_POPUP_V6_DESTROY_SINCE_VERSION** 1
- #define **ZXDG_POPUP_V6_GRAB_SINCE_VERSION** 1

14.41.1 Detailed Description

A popup surface is a short-lived, temporary surface. It can be used to implement for example menus, popovers, tooltips and other similar user interface concepts.

A popup can be made to take an explicit grab. See xdg_popup.grab for details.

When the popup is dismissed, a popup_done event will be sent out, and at the same time the surface will be unmapped. See the xdg_popup.popup_done event for details.

Explicitly destroying the xdg_popup object will also dismiss the popup and unmap the surface. Clients that want to dismiss the popup when another surface of their own is clicked should dismiss the popup using the destroy request.

The parent surface must have either the xdg_toplevel or xdg_popup surface role.

A newly created xdg_popup will be stacked on top of all previously created xdg_popup surfaces associated with the same xdg_toplevel.

The parent of an xdg_popup must be mapped (see the xdg_surface description) before the xdg_popup itself.

The x and y arguments passed when creating the popup object specify where the top left of the popup should be placed, relative to the local surface coordinates of the parent surface. See xdg_surface.get_popup. An xdg_popup must intersect with or be at least partially adjacent to its parent surface.

The client must call wl_surface.commit on the corresponding wl_surface for the xdg_popup state to take effect.

Chapter 15

Namespace Documentation

15.1 Common Namespace Reference

C++ namespace for the Common schema namespace.

Classes

- class alpha

  Class corresponding to the alpha schema type.
- class baseResources

  Class corresponding to the baseResources schema type.
- class color

  Class corresponding to the color schema type.
- class events

  Class corresponding to the events schema type.
- class font

  Class corresponding to the font schema type.
- class onAttack

  Class corresponding to the onAttack schema type.
- class onAttacked

  Class corresponding to the onAttacked schema type.
- class onClick

  Class corresponding to the onClick schema type.
- class onDestroyed

  Class corresponding to the onDestroyed schema type.
- class onEnter

  Class corresponding to the onEnter schema type.
- class onLeave

  Class corresponding to the onLeave schema type.
- class position

  Class corresponding to the position schema type.
- class preloadResources

  Class corresponding to the preloadResources schema type.
- class resources

  Class corresponding to the resources schema type.
- class size

  Class corresponding to the size schema type.

Functions

Parsing functions for the preloadResources document root.

- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (const ::std::string &uri, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (const ::std::string &uri, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (const ::std::string &uri, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (::std::istream &is, ::xercesc::D↩ OMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (::std::istream &is, const ::std↩ ::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (::std::istream &is, const ::std↩ ::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (::std::istream &is, const ::std↩ ::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (::xercesc::InputSource &is, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::preloadResources > preloadResources_ (::xml_schema::dom::unique_↩ ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the resources document root.

- ::std::unique_ptr< ::Common::resources > resources_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

Parse a URI or a local file.

- ::std::unique_ptr< ::Common::resources > resources_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::resources > resources_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::resources > resources_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::resources > resources_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::resources > resources_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::resources > resources_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::resources > resources_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::resources > resources_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::resources > resources_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::resources > resources_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::resources > resources_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::resources > resources_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::resources > resources_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the events document root.

- ::std::unique_ptr< ::Common::events > events_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::events > events_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::events > events_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::events > events_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::events > events_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

Parse a standard input stream with an error handler.

- ::std::unique_ptr< ::Common::events > events_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::events > events_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::events > events_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::events > events_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::events > events_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::events > events_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::events > events_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::events > events_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::events > events_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the position document root.

- ::std::unique_ptr< ::Common::position > position_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::position > position_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::position > position_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::position > position_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::position > position_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::position > position_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::position > position_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::position > position_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.

- ::std::unique_ptr< ::Common::position > position_ (::std::istream &is, const ::std::string &id, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::position > position_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::position > position_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::position > position_ (::xercesc::InputSource &is, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::position > position_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::position > position_ (::xml_schema::dom::unique_ptr< ::xercesc::DOM↩ Document > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the size document root.

- ::std::unique_ptr< ::Common::size > size_ (const ::std::string &uri, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::size > size_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::size > size_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::size > size_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::size > size_ (::std::istream &is, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::size > size_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::size > size_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::size > size_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::size > size_ (::std::istream &is, const ::std::string &id, ::xercesc::DOM↩ ErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::size > size_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::size > size_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::size > size_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

- ::std::unique_ptr< ::Common::size > size_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::size > size_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the color document root.

- ::std::unique_ptr< ::Common::color > color_ (const ::std::string &uri, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::color > color_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::color > color_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::color > color_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::color > color_ (::std::istream &is, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::color > color_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::color > color_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::color > color_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::color > color_ (::std::istream &is, const ::std::string &id, ::xercesc::DOM↩ ErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::color > color_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::color > color_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::color > color_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::color > color_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::color > color_ (::xml_schema::dom::unique_ptr< ::xercesc::DOM↩ Document > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the font document root.

- ::std::unique_ptr< ::Common::font > font_ (const ::std::string &uri, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.

- ::std::unique_ptr< ::Common::font > font_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::font > font_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::font > font_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream.
- ::std::unique_ptr< ::Common::font > font_ (::std::istream &is, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::font > font_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::font > font_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::font > font_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::font > font_ (::std::istream &is, const ::std::string &id, ::xercesc::DOM↩ ErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::font > font_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())
  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::font > font_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::font > font_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::font > font_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::font > font_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a Xerces-C++ DOM document.

15.1.1 Detailed Description

C++ namespace for the Common schema namespace.

15.1.2 Function Documentation

15.1.2.1 color_() [1/14]

```
std::unique_ptr<::Common::color > Common::color_ (
            ::std::istream & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.2 color_() [2/14]

```
std::unique_ptr<::Common::color > Common::color_ (
            ::std::istream & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.3 color_() [3/14]

```
std::unique_ptr<::Common::color > Common::color_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.4 color_() [4/14]

```
std::unique_ptr<::Common::color > Common::color_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.5 color_() [5/14]

```
std::unique_ptr<::Common::color > Common::color_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.6 color_() [6/14]

```
std::unique_ptr<::Common::color > Common::color_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.1.2.7 color_() [7/14]

```
std::unique_ptr<::Common::color > Common::color_ (
              ::xercesc::InputSource & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.8 color_() [8/14]

```
std::unique_ptr<::Common::color > Common::color_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.9 color_() [9/14]

```
std::unique_ptr<::Common::color > Common::color_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.10 color_() [10/14]

```
std::unique_ptr<::Common::color > Common::color_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.1.2.11 color_() [11/14]

```
std::unique_ptr<::Common::color > Common::color_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.12 color_() [12/14]

```
std::unique_ptr<::Common::color > Common::color_ (
            const ::std::string & uri,
```

```
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.13 color_() [13/14]

```
std::unique_ptr<::Common::color > Common::color_ (
            const ::std::string & uri,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.14 color_() [14/14]

```
std::unique_ptr<::Common::color > Common::color_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.1.2.15 events_() [1/14]

```
std::unique_ptr<::Common::events > Common::events_ (
              ::std::istream & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.16 events_() [2/14]

```
std::unique_ptr<::Common::events > Common::events_ (
              ::std::istream & is,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.17 events_() [3/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.18 events_() [4/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.19 events_() [5/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.20 events_() [6/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.1.2.21 events_() [7/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            ::xercesc::InputSource & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.22 events_() [8/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.23 events_() [9/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.24 events_() [10/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.1.2.25 events_() [11/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.26 events_() [12/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            const ::std::string & uri,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.27 events_() [13/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            const ::std::string & uri,
```

```
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.28 events_() [14/14]

```
std::unique_ptr<::Common::events > Common::events_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.1.2.29 font_() [1/14]

```
std::unique_ptr<::Common::font > Common::font_ (
            ::std::istream & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.30 font_() [2/14]

```
std::unique_ptr<::Common::font > Common::font_ (
            ::std::istream & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.31 font_() [3/14]

```
std::unique_ptr<::Common::font > Common::font_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.32 font_() [4/14]

```
std::unique_ptr<::Common::font > Common::font_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.33 font_() [5/14]

```
std::unique_ptr<::Common::font > Common::font_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.34 font_() [6/14]

```
std::unique_ptr<::Common::font > Common::font_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.1.2.35 font_() [7/14]

```
std::unique_ptr<::Common::font > Common::font_ (
              ::xercesc::InputSource & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.36 font_() [8/14]

```
std::unique_ptr<::Common::font > Common::font_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.37 font_() [9/14]

```
std::unique_ptr<::Common::font > Common::font_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.38 font_() [10/14]

```
std::unique_ptr<::Common::font > Common::font_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.1.2.39 font_() [11/14]

```
std::unique_ptr<::Common::font > Common::font_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.40 font_() [12/14]

```
std::unique_ptr<::Common::font > Common::font_ (
            const ::std::string & uri,
```

```
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.41 font_() [13/14]

```
std::unique_ptr<::Common::font > Common::font_ (
            const ::std::string & uri,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.42 font_() [14/14]

```
std::unique_ptr<::Common::font > Common::font_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.1.2.43 position_() [1/14]

```
std::unique_ptr<::Common::position > Common::position_ (
              ::std::istream & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.44 position_() [2/14]

```
std::unique_ptr<::Common::position > Common::position_ (
              ::std::istream & is,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.45 position_() [3/14]

```
std::unique_ptr<::Common::position > Common::position_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.46 position_() [4/14]

```
std::unique_ptr<::Common::position > Common::position_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.47 position_() [5/14]

```
std::unique_ptr<::Common::position > Common::position_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.48 position_() [6/14]

```
std::unique_ptr<::Common::position > Common::position_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.1.2.49 position_() [7/14]

```
std::unique_ptr<::Common::position > Common::position_ (
            ::xercesc::InputSource & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.50 position_() [8/14]

```
std::unique_ptr<::Common::position > Common::position_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.51 position_() [9/14]

```
std::unique_ptr<::Common::position > Common::position_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.52 position_() [10/14]

```
std::unique_ptr<::Common::position > Common::position_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.1.2.53 position_() [11/14]

```
std::unique_ptr<::Common::position > Common::position_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.54 position_() [12/14]

```
std::unique_ptr<::Common::position > Common::position_ (
            const ::std::string & uri,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.55 position_() [13/14]

```
std::unique_ptr<::Common::position > Common::position_ (
            const ::std::string & uri,
```

```
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.56 position_() [14/14]

```
std::unique_ptr<::Common::position > Common::position_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.1.2.57 preloadResources_() [1/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
            ::std::istream & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.58 preloadResources_() [2/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
            ::std::istream & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.59 preloadResources_() [3/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.60 preloadResources_() [4/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.61 preloadResources_() [5/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.62 preloadResources_() [6/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.1.2.63 preloadResources_() [7/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
            ::xercesc::InputSource & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.64 preloadResources_() [8/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.65 preloadResources_() [9/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.66 preloadResources_() [10/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.1.2.67 preloadResources_() [11/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.68 preloadResources_() [12/14]

```
std::unique_ptr<::Common::preloadResources > Common::preloadResources_ (
            const ::std::string & uri,
```

```
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.69 preloadResources_() [13/14]

```
std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (
            const ::std::string & uri,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.70 preloadResources_() [14/14]

```
std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.1.2.71 resources_() [1/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
              ::std::istream & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.72 resources_() [2/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
              ::std::istream & is,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.73 resources_() [3/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.74 resources_() [4/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.75 resources_() [5/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.76 resources_() [6/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.1.2.77 resources_() [7/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
              ::xercesc::InputSource & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.78 resources_() [8/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
              ::xercesc::InputSource & is,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.79 resources_() [9/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.80 resources_() [10/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.1.2.81 resources_() [11/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.82 resources_() [12/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
            const ::std::string & uri,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.83 resources_() [13/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
            const ::std::string & uri,
```

```
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.84 resources_() [14/14]

```
std::unique_ptr<::Common::resources > Common::resources_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.1.2.85 size_() [1/14]

```
std::unique_ptr<::Common::size > Common::size_ (
            ::std::istream & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.86 size_() [2/14]

```
std::unique_ptr<::Common::size > Common::size_ (
            ::std::istream & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.87 size_() [3/14]

```
std::unique_ptr<::Common::size > Common::size_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.88 size_() [4/14]

```
std::unique_ptr<::Common::size > Common::size_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.89 size_() [5/14]

```
std::unique_ptr<::Common::size > Common::size_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.1.2.90 size_() [6/14]

```
std::unique_ptr<::Common::size > Common::size_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.1.2.91 size_() [7/14]

```
std::unique_ptr<::Common::size > Common::size_ (
              ::xercesc::InputSource & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.92 size_() [8/14]

```
std::unique_ptr<::Common::size > Common::size_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.93 size_() [9/14]

```
std::unique_ptr<::Common::size > Common::size_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.94 size_() [10/14]

```
std::unique_ptr<::Common::size > Common::size_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.1.2.95 size_() [11/14]

```
std::unique_ptr<::Common::size > Common::size_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.96 size_() [12/14]

```
std::unique_ptr<::Common::size > Common::size_ (
            const ::std::string & uri,
```

```
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.1.2.97 size_() [13/14]

```
std::unique_ptr<::Common::size > Common::size_ (
            const ::std::string & uri,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.1.2.98 size_() [14/14]

```
std::unique_ptr<::Common::size > Common::size_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.2 Components Namespace Reference

C++ namespace for the Components schema namespace.

Classes

- class bodyShape

  Class corresponding to the bodyShape schema type.
- class bodyType

  Class corresponding to the bodyType schema type.
- class box

  Class corresponding to the box schema type.
- class bulletComponent

  Class corresponding to the bulletComponent schema type.
- class characterComponent

  Class corresponding to the characterComponent schema type.
- class circle

  Class corresponding to the circle schema type.
- class component

  Class corresponding to the component schema type.
- class componentName

  Class corresponding to the componentName schema type.
- class explosionCrate

  Class corresponding to the explosionCrate schema type.
- class floatCap

  Class corresponding to the floatCap schema type.
- class nextLevelComponent

  Class corresponding to the nextLevelComponent schema type.
- class physicsComponent

  Class corresponding to the physicsComponent schema type.
- class renderComponent

  Class corresponding to the renderComponent schema type.
- class transformComponent

  Class corresponding to the transformComponent schema type.

15.2.1 Detailed Description

C++ namespace for the Components schema namespace.

15.3 GameResources Namespace Reference

C++ namespace for the GameResources schema namespace.

Classes

- class baseGameResource

  Class corresponding to the baseGameResource schema type.
- class level

  Class corresponding to the level schema type.
- class levels

  Class corresponding to the levels schema type.
- class music

  Class corresponding to the music schema type.
- class music1

  Class corresponding to the music1 schema type.
- class objectList

  Class corresponding to the objectList schema type.
- class objectLists

  Class corresponding to the objectLists schema type.
- class pool

  Class corresponding to the pool schema type.
- class resources

  Class corresponding to the resources schema type.
- class scene

  Class corresponding to the scene schema type.
- class scenes

  Class corresponding to the scenes schema type.
- class sound

  Class corresponding to the sound schema type.
- class sounds

  Class corresponding to the sounds schema type.
- class sprite

  Class corresponding to the sprite schema type.
- class sprites

  Class corresponding to the sprites schema type.
- class texture

  Class corresponding to the texture schema type.
- class textures

  Class corresponding to the textures schema type.

Functions

Parsing functions for the resources document root.

- ::std::unique_ptr< ::GameResources::resources > resources_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::GameResources::resources > resources_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::GameResources::resources > resources_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::GameResources::resources > resources_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::GameResources::resources > resources_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::GameResources::resources > resources_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::GameResources::resources > resources_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::GameResources::resources > resources_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::GameResources::resources > resources_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::GameResources::resources > resources_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::GameResources::resources > resources_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::GameResources::resources > resources_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::GameResources::resources > resources_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::GameResources::resources > resources_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

15.3.1 Detailed Description

C++ namespace for the GameResources schema namespace.

15.3.2 Function Documentation

15.3.2.1 resources_() [1/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            ::std::istream & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.3.2.2 resources_() [2/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            ::std::istream & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.3.2.3 resources_() [3/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.3.2.4 resources_() [4/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.3.2.5 resources_() [5/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.3.2.6 resources_() [6/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.3.2.7 resources_() [7/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            ::xercesc::InputSource & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.3.2.8 resources_() [8/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.3.2.9 resources_() [9/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            ::xercesc::InputSource & is,
```

```
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.3.2.10 resources_() [10/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.3.2.11 resources_() [11/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.3.2.12 resources_() [12/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            const ::std::string & uri,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.3.2.13 resources_() [13/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            const ::std::string & uri,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.3.2.14 resources_() [14/14]

```
std::unique_ptr<::GameResources::resources > GameResources::resources_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.4 LevelResources Namespace Reference

C++ namespace for the LevelResources schema namespace.

Classes

- class level

  Class corresponding to the level schema type.

Functions

Parsing functions for the level document root.

- ::std::unique_ptr< ::LevelResources::level > level_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::LevelResources::level > level_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::LevelResources::level > level_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::LevelResources::level > level_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

Parse a standard input stream.

- ::std::unique_ptr< ::LevelResources::level > level_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::LevelResources::level > level_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::LevelResources::level > level_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::LevelResources::level > level_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::LevelResources::level > level_ (::std::istream &is, const ::std::string &id, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::LevelResources::level > level_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::LevelResources::level > level_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::LevelResources::level > level_ (::xercesc::InputSource &is, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::LevelResources::level > level_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::LevelResources::level > level_ (::xml_schema::dom::unique_ptr< ::xercesc::DOM↩ Document > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

15.4.1 Detailed Description

C++ namespace for the LevelResources schema namespace.

15.4.2 Function Documentation

15.4.2.1 level_() [1/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
            ::std::istream & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.4.2.2 level_() [2/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
            ::std::istream & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.4.2.3 level_() [3/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.4.2.4 level_() [4/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.4.2.5 level_() [5/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.4.2.6 level_() [6/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.4.2.7 level_() [7/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
              ::xercesc::InputSource & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.4.2.8 level_() [8/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.4.2.9 level_() [9/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.4.2.10 level_() [10/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.4.2.11 level_() [11/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.4.2.12 level_() [12/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
            const ::std::string & uri,
```

```
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.4.2.13 level_() [13/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
            const ::std::string & uri,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.4.2.14 level_() [14/14]

```
std::unique_ptr<::LevelResources::level > LevelResources::level_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.5 Menu Namespace Reference

C++ namespace for the Menu schema namespace.

Classes

- class box

  Class corresponding to the box schema type.
- class boxes

  Class corresponding to the boxes schema type.
- class button

  Class corresponding to the button schema type.
- class buttons

  Class corresponding to the buttons schema type.
- class image

  Class corresponding to the image schema type.
- class images

  Class corresponding to the images schema type.
- class menu

  Class corresponding to the menu schema type.
- class resources

  Class corresponding to the resources schema type.
- class text

  Class corresponding to the text schema type.
- class text1

  Class corresponding to the text1 schema type.
- class texts

  Class corresponding to the texts schema type.

Functions

Parsing functions for the menu document root.

- ::std::unique_ptr< ::Menu::menu > menu_ (const ::std::string &uri, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())
  Parse a URI or a local file.
- ::std::unique_ptr< ::Menu::menu > menu_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Menu::menu > menu_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Menu::menu > menu_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream.
- ::std::unique_ptr< ::Menu::menu > menu_ (::std::istream &is, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Menu::menu > menu_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Menu::menu > menu_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Menu::menu > menu_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Menu::menu > menu_ (::std::istream &is, const ::std::string &id, ::xercesc::DOM↩ ErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Menu::menu > menu_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())
  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Menu::menu > menu_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Menu::menu > menu_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Menu::menu > menu_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Menu::menu > menu_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())
  Parse a Xerces-C++ DOM document.

15.5.1 Detailed Description

C++ namespace for the Menu schema namespace.

15.5.2 Function Documentation

15.5.2.1 menu_() [1/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
              ::std::istream & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.5.2.2 menu_() [2/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
              ::std::istream & is,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.5.2.3 menu_() [3/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
              ::std::istream & is,
```

```
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.5.2.4 menu_() [4/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.5.2.5 menu_() [5/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
            ::std::istream & is,
            const ::std::string & id,
```

```
::xml_schema::error_handler & eh,
::xml_schema::flags f = 0,
const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.5.2.6 menu_() [6/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.5.2.7 menu_() [7/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
              ::xercesc::InputSource & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.5.2.8 menu_() [8/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.5.2.9 menu_() [9/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.5.2.10 menu_() [10/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.5.2.11 menu_() [11/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.5.2.12 menu_() [12/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
            const ::std::string & uri,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.5.2.13 menu_() [13/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
            const ::std::string & uri,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.5.2.14 menu_() [14/14]

```
std::unique_ptr<::Menu::menu > Menu::menu_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.6 Objects Namespace Reference

C++ namespace for the Objects schema namespace.

Classes

- class components

  Class corresponding to the components schema type.
- class object

  Class corresponding to the object schema type.
- class objectList

  Class corresponding to the objectList schema type.

Functions

Parsing functions for the objectList document root.

- ::std::unique_ptr< ::Objects::objectList > objectList_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

Parse a standard input stream with a resource id and an error handler.

- ::std::unique_ptr< ::Objects::objectList > objectList_ (::std::istream &is, const ::std::string &id, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (::xercesc::InputSource &is, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (const ::xercesc::DOMDocument &d, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Objects::objectList > objectList_ (::xml_schema::dom::unique_ptr< ::xercesc::DOM↩ Document > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

15.6.1 Detailed Description

C++ namespace for the Objects schema namespace.

15.6.2 Function Documentation

15.6.2.1 objectList_() [1/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            ::std::istream & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.6.2.2 objectList_() [2/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            ::std::istream & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.6.2.3 objectList_() [3/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.6.2.4 objectList_() [4/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            ::std::istream & is,
            const ::std::string & id,
```

```
::xercesc::DOMErrorHandler & eh,
::xml_schema::flags f = 0,
const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.6.2.5 objectList_() [5/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.6.2.6 objectList_() [6/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.6.2.7 objectList_() [7/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            ::xercesc::InputSource & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.6.2.8 objectList_() [8/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.6.2.9 objectList_() [9/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.6.2.10 objectList_() [10/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.6.2.11 objectList_() [11/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.6.2.12 objectList_() [12/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            const ::std::string & uri,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.6.2.13 objectList_() [13/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            const ::std::string & uri,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.6.2.14 objectList_() [14/14]

```
std::unique_ptr<::Objects::objectList > Objects::objectList_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.7 Sentry Namespace Reference

C++ namespace for the Sentry schema namespace.

Classes

- class sentry

  Class corresponding to the sentry schema type.

Functions

Parsing functions for the sentry document root.

- ::std::unique_ptr< ::Sentry::sentry > sentry_ (const ::std::string &uri, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (::std::istream &is, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (::std::istream &is, const ::std::string &id, ::xercesc::DOM↩ ErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Sentry::sentry > sentry_ (::xml_schema::dom::unique_ptr< ::xercesc::DOM↩ Document > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

15.7.1 Detailed Description

C++ namespace for the Sentry schema namespace.

15.7.2 Function Documentation

15.7.2.1 sentry_() [1/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            ::std::istream & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.7.2.2 sentry_() [2/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            ::std::istream & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.7.2.3 sentry_() [3/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.7.2.4 sentry_() [4/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.7.2.5 sentry_() [5/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.7.2.6 sentry_() [6/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.7.2.7 sentry_() [7/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            ::xercesc::InputSource & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.7.2.8 sentry_() [8/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.7.2.9 sentry_() [9/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            ::xercesc::InputSource & is,
```

```
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.7.2.10 sentry_() [10/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.7.2.11 sentry_() [11/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.7.2.12 sentry_() [12/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            const ::std::string & uri,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.7.2.13 sentry_() [13/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            const ::std::string & uri,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.7.2.14 sentry_() [14/14]

```
std::unique_ptr<::Sentry::sentry > Sentry::sentry_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.8 Walls Namespace Reference

C++ namespace for the Walls schema namespace.

Classes

- class powers

  Class corresponding to the powers schema type.
- class pricing

  Class corresponding to the pricing schema type.
- class upgrade

  Class corresponding to the upgrade schema type.
- class wall

  Class corresponding to the wall schema type.
- class walls

  Class corresponding to the walls schema type.

Functions

Parsing functions for the walls document root.

- ::std::unique_ptr< ::Walls::walls > walls_ (const ::std::string &uri, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Walls::walls > walls_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Walls::walls > walls_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::walls > walls_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Walls::walls > walls_ (::std::istream &is, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Walls::walls > walls_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::walls > walls_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Walls::walls > walls_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Walls::walls > walls_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::walls > walls_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Walls::walls > walls_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Walls::walls > walls_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::walls > walls_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Walls::walls > walls_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the wall document root.

- ::std::unique_ptr< ::Walls::wall > wall_ (const ::std::string &uri, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Walls::wall > wall_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Walls::wall > wall_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.

- ::std::unique_ptr< ::Walls::wall > wall_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Walls::wall > wall_ (::std::istream &is, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Walls::wall > wall_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::wall > wall_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Walls::wall > wall_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Walls::wall > wall_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::wall > wall_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Walls::wall > wall_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Walls::wall > wall_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::wall > wall_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Walls::wall > wall_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

15.8.1 Detailed Description

C++ namespace for the Walls schema namespace.

15.8.2 Function Documentation

15.8.2.1 wall_() [1/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
            ::std::istream & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.2 wall_() [2/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
              ::std::istream & is,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.3 wall_() [3/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
              ::std::istream & is,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.8.2.4 wall_() [4/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.8.2.5 wall_() [5/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.8.2.6 wall_() [6/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.8.2.7 wall_() [7/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
              ::xercesc::InputSource & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.8 wall_() [8/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.9 wall_() [9/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.8.2.10 wall_() [10/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
              ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.8.2.11 wall_() [11/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
              const ::std::string & uri,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.12 wall_() [12/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
              const ::std::string & uri,
```

```
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.13 wall_() [13/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
            const ::std::string & uri,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.8.2.14 wall_() [14/14]

```
std::unique_ptr<::Walls::wall > Walls::wall_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.8.2.15 walls_() [1/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
              ::std::istream & is,
              ::xercesc::DOMErrorHandler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.16 walls_() [2/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
              ::std::istream & is,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with an error handler.

Parameters

is	A standrad input stream.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.17 walls_() [3/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
            ::std::istream & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream.

Parameters

is	A standrad input stream.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.8.2.18 walls_() [4/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
            ::std::istream & is,
            const ::std::string & id,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.8.2.19 walls_() [5/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::error_handler & eh,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id and an error handler.

Parameters

is	A standrad input stream.
id	A resource id.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function reports parsing errors by calling the error handler.

15.8.2.20 walls_() [6/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
              ::std::istream & is,
              const ::std::string & id,
              ::xml_schema::flags f = 0,
              const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a standard input stream with a resource id.

Parameters

is	A standrad input stream.
id	A resource id.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

The resource id is used to identify the document being parsed in diagnostics as well as to resolve relative paths.

This function uses exceptions to report parsing errors.

15.8.2.21 walls_() [7/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
            ::xercesc::InputSource & is,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

Parameters

is	A Xerces-C++ input source.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.22 walls_() [8/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
            ::xercesc::InputSource & is,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source with an error handler.

Parameters

is	A Xerces-C++ input source.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.23 walls_() [9/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
            ::xercesc::InputSource & is,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ input source.

Parameters

is	A Xerces-C++ input source.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.8.2.24 walls_() [10/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
            ::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A pointer to the Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function is normally used together with the keep_dom and own_dom parsing flags to assign ownership of the DOM document to the object model.

15.8.2.25 walls_() [11/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
            const ::std::string & uri,
            ::xercesc::DOMErrorHandler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with a Xerces-C++ DOM error handler.

Parameters

uri	A URI or a local file name.
eh	A Xerces-C++ DOM error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.26 walls_() [12/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
            const ::std::string & uri,
            ::xml_schema::error_handler & eh,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file with an error handler.

Parameters

uri	A URI or a local file name.
eh	An error handler.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function reports parsing errors by calling the error handler.

15.8.2.27 walls_() [13/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
            const ::std::string & uri,
```

```
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a URI or a local file.

Parameters

uri	A URI or a local file name.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

This function uses exceptions to report parsing errors.

15.8.2.28 walls_() [14/14]

```
std::unique_ptr<::Walls::walls > Walls::walls_ (
            const ::xercesc::DOMDocument & d,
            ::xml_schema::flags f = 0,
            const ::xml_schema::properties & p = ::xml_schema::properties() )
```

Parse a Xerces-C++ DOM document.

Parameters

d	A Xerces-C++ DOM document.
f	Parsing flags.
p	Parsing properties.

Returns

A pointer to the root of the object model.

15.9 xml_schema Namespace Reference

C++ namespace for the http://www.w3.org/2001/XMLSchema schema namespace.

Namespaces

- dom

  DOM interaction.

Typedefs

- typedef ::xsd::cxx::tree::type type

  C++ type corresponding to the anyType XML Schema built-in type.
- typedef ::xsd::cxx::tree::simple_type< char, type > simple_type

  C++ type corresponding to the anySimpleType XML Schema built-in type.
- typedef ::xsd::cxx::tree::type container

  Alias for the anyType type.
- typedef signed char byte

  C++ type corresponding to the byte XML Schema built-in type.
- typedef unsigned char unsigned_byte

  C++ type corresponding to the unsignedByte XML Schema built-in type.
- typedef short short_

  C++ type corresponding to the short XML Schema built-in type.
- typedef unsigned short unsigned_short

  C++ type corresponding to the unsignedShort XML Schema built-in type.
- typedef int int_

  C++ type corresponding to the int XML Schema built-in type.
- typedef unsigned int unsigned_int

  C++ type corresponding to the unsignedInt XML Schema built-in type.
- typedef long long long_

  C++ type corresponding to the long XML Schema built-in type.
- typedef unsigned long long unsigned_long

  C++ type corresponding to the unsignedLong XML Schema built-in type.
- typedef long long integer

  C++ type corresponding to the integer XML Schema built-in type.
- typedef long long non_positive_integer

  C++ type corresponding to the nonPositiveInteger XML Schema built-in type.
- typedef unsigned long long non_negative_integer

  C++ type corresponding to the nonNegativeInteger XML Schema built-in type.
- typedef unsigned long long positive_integer

  C++ type corresponding to the positiveInteger XML Schema built-in type.
- typedef long long negative_integer

  C++ type corresponding to the negativeInteger XML Schema built-in type.
- typedef bool boolean

  C++ type corresponding to the boolean XML Schema built-in type.
- typedef float float_

  C++ type corresponding to the float XML Schema built-in type.
- typedef double double_

  C++ type corresponding to the double XML Schema built-in type.
- typedef double decimal

  C++ type corresponding to the decimal XML Schema built-in type.
- typedef ::xsd::cxx::tree::string< char, simple_type > string

  C++ type corresponding to the string XML Schema built-in type.
- typedef ::xsd::cxx::tree::normalized_string< char, string > normalized_string

  C++ type corresponding to the normalizedString XML Schema built-in type.
- typedef ::xsd::cxx::tree::token< char, normalized_string > token

  C++ type corresponding to the token XML Schema built-in type.
- typedef ::xsd::cxx::tree::name< char, token > name

  C++ type corresponding to the Name XML Schema built-in type.
- typedef ::xsd::cxx::tree::nmtoken< char, token > nmtoken

C++ type corresponding to the NMTOKEN XML Schema built-in type.

- typedef ::xsd::cxx::tree::nmtokens< char, simple_type, nmtoken > nmtokens

  C++ type corresponding to the NMTOKENS XML Schema built-in type.
- typedef ::xsd::cxx::tree::ncname< char, name > ncname

  C++ type corresponding to the NCName XML Schema built-in type.
- typedef ::xsd::cxx::tree::language< char, token > language

  C++ type corresponding to the language XML Schema built-in type.
- typedef ::xsd::cxx::tree::id< char, ncname > id

  C++ type corresponding to the ID XML Schema built-in type.
- typedef ::xsd::cxx::tree::idref< char, ncname, type > idref

  C++ type corresponding to the IDREF XML Schema built-in type.
- typedef ::xsd::cxx::tree::idrefs< char, simple_type, idref > idrefs

  C++ type corresponding to the IDREFS XML Schema built-in type.
- typedef ::xsd::cxx::tree::uri< char, simple_type > uri

  C++ type corresponding to the anyURI XML Schema built-in type.
- typedef ::xsd::cxx::tree::qname< char, simple_type, uri, ncname > qname

  C++ type corresponding to the QName XML Schema built-in type.
- typedef ::xsd::cxx::tree::buffer< char > buffer

  Binary buffer type.
- typedef ::xsd::cxx::tree::base64_binary< char, simple_type > base64_binary

  C++ type corresponding to the base64Binary XML Schema built-in type.
- typedef ::xsd::cxx::tree::hex_binary< char, simple_type > hex_binary

  C++ type corresponding to the hexBinary XML Schema built-in type.
- typedef ::xsd::cxx::tree::time_zone time_zone

  Time zone type.
- typedef ::xsd::cxx::tree::date< char, simple_type > date

  C++ type corresponding to the date XML Schema built-in type.
- typedef ::xsd::cxx::tree::date_time< char, simple_type > date_time

  C++ type corresponding to the dateTime XML Schema built-in type.
- typedef ::xsd::cxx::tree::duration< char, simple_type > duration

  C++ type corresponding to the duration XML Schema built-in type.
- typedef ::xsd::cxx::tree::gday< char, simple_type > gday

  C++ type corresponding to the gDay XML Schema built-in type.
- typedef ::xsd::cxx::tree::gmonth< char, simple_type > gmonth

  C++ type corresponding to the gMonth XML Schema built-in type.
- typedef ::xsd::cxx::tree::gmonth_day< char, simple_type > gmonth_day

  C++ type corresponding to the gMonthDay XML Schema built-in type.
- typedef ::xsd::cxx::tree::gyear< char, simple_type > gyear

  C++ type corresponding to the gYear XML Schema built-in type.
- typedef ::xsd::cxx::tree::gyear_month< char, simple_type > gyear_month

  C++ type corresponding to the gYearMonth XML Schema built-in type.
- typedef ::xsd::cxx::tree::time< char, simple_type > time

  C++ type corresponding to the time XML Schema built-in type.
- typedef ::xsd::cxx::tree::entity< char, ncname > entity

  C++ type corresponding to the ENTITY XML Schema built-in type.
- typedef ::xsd::cxx::tree::entities< char, simple_type, entity > entities

  C++ type corresponding to the ENTITIES XML Schema built-in type.
- typedef ::xsd::cxx::tree::content_order content_order

  Content order sequence entry.
- typedef ::xsd::cxx::tree::flags flags

  Parsing and serialization flags.

- typedef ::xsd::cxx::tree::properties< char > properties

  Parsing properties.
- typedef ::xsd::cxx::tree::severity severity

  Error severity.
- typedef ::xsd::cxx::tree::error< char > error

  Error condition.
- typedef ::xsd::cxx::tree::diagnostics< char > diagnostics

  List of error conditions.
- typedef ::xsd::cxx::tree::exception< char > exception

  Root of the C++/Tree exception hierarchy.
- typedef ::xsd::cxx::tree::bounds< char > bounds

  Exception indicating that the size argument exceeds the capacity argument.
- typedef ::xsd::cxx::tree::duplicate_id< char > duplicate_id

  Exception indicating that a duplicate ID value was encountered in the object model.
- typedef ::xsd::cxx::tree::parsing< char > parsing

  Exception indicating a parsing failure.
- typedef ::xsd::cxx::tree::expected_element< char > expected_element

  Exception indicating that an expected element was not encountered.
- typedef ::xsd::cxx::tree::unexpected_element< char > unexpected_element

  Exception indicating that an unexpected element was encountered.
- typedef ::xsd::cxx::tree::expected_attribute< char > expected_attribute

  Exception indicating that an expected attribute was not encountered.
- typedef ::xsd::cxx::tree::unexpected_enumerator< char > unexpected_enumerator

  Exception indicating that an unexpected enumerator was encountered.
- typedef ::xsd::cxx::tree::expected_text_content< char > expected_text_content

  Exception indicating that the text content was expected for an element.
- typedef ::xsd::cxx::tree::no_prefix_mapping< char > no_prefix_mapping

  Exception indicating that a prefix-namespace mapping was not provided.
- typedef ::xsd::cxx::xml::error_handler< char > error_handler

  Error handler callback interface.

15.9.1 Detailed Description

C++ namespace for the http://www.w3.org/2001/XMLSchema schema namespace.

15.10 xml_schema::dom Namespace Reference

DOM interaction.

Variables

- const XMLCh *const tree_node_key = ::xsd::cxx::tree::user_data_keys::node

  DOM user data key for back pointers to tree nodes.

15.10.1 Detailed Description

DOM interaction.

Chapter 16

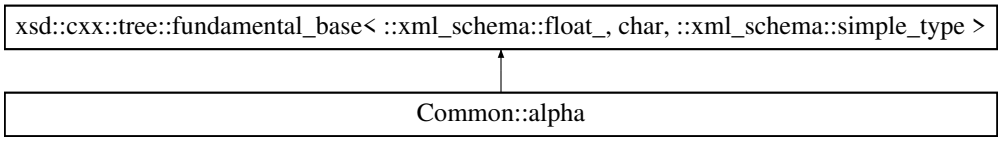
Class Documentation

16.1 Common::alpha Class Reference

Class corresponding to the alpha schema type.

`#include <common.hxx>`

Inheritance diagram for Common::alpha:



Constructors

- alpha (const ::xml_schema::float_ &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- alpha (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container ∗c=0)

  Create an instance from a DOM element.
- alpha (const ::xercesc::DOMAttr &a, ::xml_schema::flags f=0, ::xml_schema::container ∗c=0)

  Create an instance from a DOM attribute.
- alpha (const ::std::string &s, const ::xercesc::DOMElement ∗e, ::xml_schema::flags f=0, ::xml_schema::container ∗c=0)

  Create an instance from a string fragment.
- alpha (const alpha &x, ::xml_schema::flags f=0, ::xml_schema::container ∗c=0)

  Copy constructor.
- virtual alpha ∗ _clone (::xml_schema::flags f=0, ::xml_schema::container ∗c=0) const

  Copy the instance polymorphically.
- virtual ∼alpha ()

  Destructor.

16.1.1 Detailed Description

Class corresponding to the alpha schema type.

16.1.2 Constructor & Destructor Documentation

16.1.2.1 alpha() [1/4]

```
Common::alpha::alpha (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.1.2.2 alpha() [2/4]

```
Common::alpha::alpha (
            const ::xercesc::DOMAttr & a,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM attribute.

Parameters

a	A DOM attribute to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.1.2.3 alpha() [3/4]

```
Common::alpha::alpha (
            const ::std::string & s,
            const ::xercesc::DOMElement * e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a string fragment.

Parameters

s	A string fragment to extract the data from.
e	A pointer to DOM element containing the string fragment.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.1.2.4 alpha() [4/4]

```
Common::alpha::alpha (
            const alpha & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.1.3 Member Function Documentation

16.1.3.1 _clone()

```
alpha * Common::alpha::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

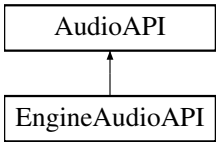
This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.2 AudioAPI Class Reference

Inheritance diagram for AudioAPI:

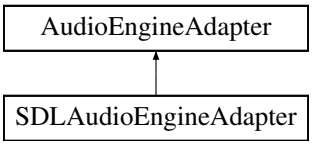


The documentation for this class was generated from the following file:

- API/Audio/AudioAPI.h

16.3 AudioEngineAdapter Class Reference

Inheritance diagram for AudioEngineAdapter:



Public Member Functions

- virtual std::vector< std::string > **getAudioNames** ()=0
- virtual void **loadInMemory** (const std::string &path, const std::string &name, AudioType type)=0
- virtual void **playFromPath** (const std::string &path, AudioType &type)=0
- virtual void **playFromMemory** (const std::string &name)=0
- virtual void **changeMasterVolume** (int volume)=0
- virtual void **changeChannelVolume** (int channel, int volume)=0
- virtual void **changeMusicVolume** (int volume)=0
- virtual int **getChannelsAverageVolume** ()=0
- virtual int **getChannelVolume** (int channel)=0
- virtual int **getMusicVolume** ()=0
- virtual void **stopAudio** ()=0
- virtual void **stopMusic** ()=0
- virtual void **stopSound** (int channel)=0
- virtual void **stopSounds** ()=0
- virtual void **toggleMusic** ()=0
- virtual void **toggleSound** (int channel)=0
- virtual void **toggleSounds** ()=0

The documentation for this class was generated from the following file:

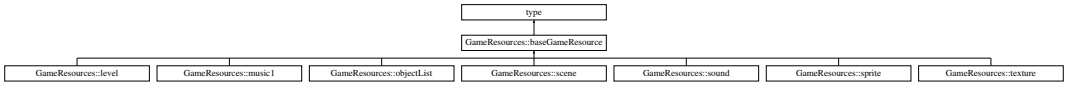
- Engine/Audio/Adapter/AudioEngineAdapter.hpp

16.4 GameResources::baseGameResource Class Reference

Class corresponding to the baseGameResource schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::baseGameResource:



name

Accessor and modifier functions for the name required element.

- typedef ::xml_schema::string name_type

  Element type.
- typedef ::xsd::cxx::tree::traits< name_type, char > name_traits

  Element traits type.
- const name_type & name () const

  Return a read-only (constant) reference to the element.
- name_type & name ()

  Return a read-write reference to the element.
- void name (const name_type &x)

  Set the element value.
- void name (::std::unique_ptr< name_type > p)

  Set the element value without copying.

path

Accessor and modifier functions for the path required element.

- typedef ::xml_schema::string path_type

  Element type.
- typedef ::xsd::cxx::tree::traits< path_type, char > path_traits

  Element traits type.
- const path_type & path () const

  Return a read-only (constant) reference to the element.
- path_type & path ()

  Return a read-write reference to the element.
- void path (const path_type &x)

  Set the element value.
- void path (::std::unique_ptr< path_type > p)

  Set the element value without copying.

Constructors

- baseGameResource (const name_type &, const path_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- baseGameResource (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- baseGameResource (const baseGameResource &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual baseGameResource * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- baseGameResource & operator= (const baseGameResource &x)

  Copy assignment operator.
- virtual ~baseGameResource ()

  Destructor.

16.4.1 Detailed Description

Class corresponding to the baseGameResource schema type.

16.4.2 Constructor & Destructor Documentation

16.4.2.1 baseGameResource() [1/2]

```
GameResources::baseGameResource::baseGameResource (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.4.2.2 baseGameResource() [2/2]

```
GameResources::baseGameResource::baseGameResource (
            const baseGameResource & x,
```

```
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.4.3 Member Function Documentation

16.4.3.1 _clone()

```
baseGameResource * GameResources::baseGameResource::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

Reimplemented in GameResources::objectList, GameResources::level, GameResources::scene, GameResources::music1, GameResources::sound, GameResources::sprite, and GameResources::texture.

16.4.3.2 name() [1/4]

```
baseGameResource::name_type & GameResources::baseGameResource::name ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.4.3.3 name() [2/4]

```
const baseGameResource::name_type & GameResources::baseGameResource::name ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.4.3.4 name() [3/4]

```
void GameResources::baseGameResource::name (
            ::std::unique_ptr< name_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.4.3.5 name() [4/4]

```
void GameResources::baseGameResource::name (
            const name_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.4.3.6 operator=()

```
baseGameResource & GameResources::baseGameResource::operator= (
            const baseGameResource & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.4.3.7 path() [1/4]

```
baseGameResource::path_type & GameResources::baseGameResource::path ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.4.3.8 path() [2/4]

```
const baseGameResource::path_type & GameResources::baseGameResource::path ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.4.3.9 path() [3/4]

```
void GameResources::baseGameResource::path (
            ::std::unique_ptr< path_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.4.3.10 path() [4/4]

```
void GameResources::baseGameResource::path (
            const path_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

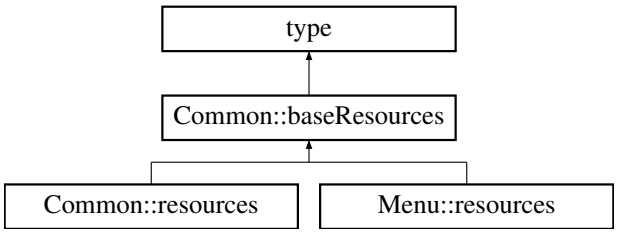
- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.5 Common::baseResources Class Reference

Class corresponding to the baseResources schema type.

`#include <common.hxx>`

Inheritance diagram for Common::baseResources:



default

Accessor and modifier functions for the default required element.

- typedef ::xml_schema::string default_type

  Element type.
- typedef ::xsd::cxx::tree::traits< default_type, char > default_traits

  Element traits type.
- const default_type & default_ () const

  Return a read-only (constant) reference to the element.
- default_type & default_ ()

  Return a read-write reference to the element.
- void default_ (const default_type &x)

  Set the element value.
- void default_ (::std::unique_ptr< default_type > p)

  Set the element value without copying.

Constructors

- baseResources (const default_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- baseResources (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- baseResources (const baseResources &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual baseResources * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- baseResources & operator= (const baseResources &x)

  Copy assignment operator.
- virtual ~baseResources ()

  Destructor.

16.5.1 Detailed Description

Class corresponding to the baseResources schema type.

16.5.2 Constructor & Destructor Documentation

16.5.2.1 baseResources() [1/2]

```
Common::baseResources::baseResources (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.5.2.2 baseResources() [2/2]

```
Common::baseResources::baseResources (
            const baseResources & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.5.3 Member Function Documentation

16.5.3.1 _clone()

```
baseResources * Common::baseResources::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

Reimplemented in Menu::resources, and Common::resources.

16.5.3.2 default_() [1/4]

```
baseResources::default_type & Common::baseResources::default_ ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.5.3.3 default_() [2/4]

```
const baseResources::default_type & Common::baseResources::default_ ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.5.3.4 default_() [3/4]

```
void Common::baseResources::default_ (
            ::std::unique_ptr< default_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.5.3.5 default_() [4/4]

```
void Common::baseResources::default_ (
            const default_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.5.3.6 operator=()

```
baseResources & Common::baseResources::operator= (
            const baseResources & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.6 BodyHandler Class Reference

Public Member Functions

- **BodyHandler** (const PhysicsEngineAdapter &physicsApi)
- void **update** ()
- void **eventOnWorldLocked** (const std::function< void()> &fun)
- void **eventOnBodiesHandles** (const std::function< void()> &fun)

The documentation for this class was generated from the following files:

- Engine/Physics/BodyHandler.hpp
- Engine/Physics/BodyHandler.cpp

16.7 BodyHandlerAPI Class Reference

Public Member Functions

- **BodyHandlerAPI** (const PhysicsAPI &physicsApi)
- void **update** ()
- void **eventOnWorldLocked** (const std::function< void()> &fun)
- void **eventOnBodiesHandled** (const std::function< void()> &fun)

The documentation for this class was generated from the following file:

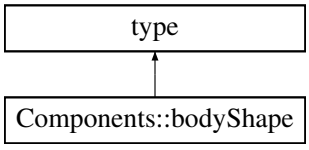
- API/Physics/BodyHandlerAPI.hpp

16.8 Components::bodyShape Class Reference

Class corresponding to the bodyShape schema type.

`#include <components.hxx>`

Inheritance diagram for Components::bodyShape:



circle

Accessor and modifier functions for the circle optional element.

- typedef ::Components::circle circle_type

  Element type.
- typedef ::xsd::cxx::tree::optional< circle_type > circle_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< circle_type, char > circle_traits

  Element traits type.
- const circle_optional & circle () const

  Return a read-only (constant) reference to the element container.
- circle_optional & circle ()

  Return a read-write reference to the element container.
- void circle (const circle_type &x)

  Set the element value.
- void circle (const circle_optional &x)

  Set the element value.
- void circle (::std::unique_ptr< circle_type > p)

  Set the element value without copying.

box

Accessor and modifier functions for the box optional element.

- typedef ::Components::box box_type

  Element type.
- typedef ::xsd::cxx::tree::optional< box_type > box_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< box_type, char > box_traits

  Element traits type.
- const box_optional & box () const

  Return a read-only (constant) reference to the element container.
- box_optional & box ()

  Return a read-write reference to the element container.
- void box (const box_type &x)

  Set the element value.
- void box (const box_optional &x)

  Set the element value.
- void box (::std::unique_ptr< box_type > p)

  Set the element value without copying.

Constructors

- bodyShape ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- bodyShape (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- bodyShape (const bodyShape &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual bodyShape * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- bodyShape & operator= (const bodyShape &x)

  Copy assignment operator.
- virtual ~bodyShape ()

  Destructor.

16.8.1 Detailed Description

Class corresponding to the bodyShape schema type.

16.8.2 Constructor & Destructor Documentation

16.8.2.1 bodyShape() [1/2]

```
Components::bodyShape::bodyShape (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.8.2.2 bodyShape() [2/2]

```
Components::bodyShape::bodyShape (
            const bodyShape & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.8.3 Member Function Documentation

16.8.3.1 _clone()

```
bodyShape * Components::bodyShape::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.8.3.2 box() [1/5]

```
bodyShape::box_optional & Components::bodyShape::box ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.8.3.3 box() [2/5]

```
const bodyShape::box_optional & Components::bodyShape::box ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.8.3.4 box() [3/5]

```
void Components::bodyShape::box (
            ::std::unique_ptr< box_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.8.3.5 box() [4/5]

```
void Components::bodyShape::box (
            const box_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.8.3.6 box() [5/5]

```
void Components::bodyShape::box (
            const box_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.8.3.7 circle() [1/5]

```
bodyShape::circle_optional & Components::bodyShape::circle ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.8.3.8 circle() [2/5]

```
const bodyShape::circle_optional & Components::bodyShape::circle ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.8.3.9 circle() [3/5]

```
void Components::bodyShape::circle (
            ::std::unique_ptr< circle_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.8.3.10 circle() [4/5]

```
void Components::bodyShape::circle (
            const circle_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.8.3.11 circle() [5/5]

```
void Components::bodyShape::circle (
            const circle_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.8.3.12 operator=()

```
bodyShape & Components::bodyShape::operator= (
            const bodyShape & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

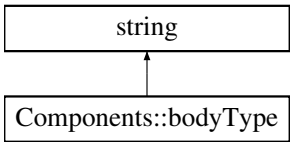
- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.9 Components::bodyType Class Reference

Class corresponding to the bodyType schema type.

`#include <components.hxx>`

Inheritance diagram for Components::bodyType:



Constructors

- bodyType ()

  Create an instance from initializers for required elements and attributes.
- bodyType (const char ∗)

  Create an instance from a C string and initializers for required elements and attributes.
- bodyType (const ::std::string &)

  Create an instance from a string andinitializers for required elements and attributes.
- bodyType (const ::xml_schema::string &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- bodyType (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container ∗c=0)

  Create an instance from a DOM element.
- bodyType (const ::xercesc::DOMAttr &a, ::xml_schema::flags f=0, ::xml_schema::container ∗c=0)

  Create an instance from a DOM attribute.
- bodyType (const ::std::string &s, const ::xercesc::DOMElement ∗e, ::xml_schema::flags f=0, ↩ ::xml_schema::container ∗c=0)

  Create an instance from a string fragment.
- bodyType (const bodyType &x, ::xml_schema::flags f=0, ::xml_schema::container ∗c=0)

  Copy constructor.
- virtual bodyType ∗ _clone (::xml_schema::flags f=0, ::xml_schema::container ∗c=0) const

  Copy the instance polymorphically.
- virtual ∼bodyType ()

  Destructor.

16.9.1 Detailed Description

Class corresponding to the bodyType schema type.

16.9.2 Constructor & Destructor Documentation

16.9.2.1 bodyType() [1/4]

```
Components::bodyType::bodyType (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.9.2.2 bodyType() [2/4]

```
Components::bodyType::bodyType (
            const ::xercesc::DOMAttr & a,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM attribute.

Parameters

a	A DOM attribute to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.9.2.3 bodyType() [3/4]

```
Components::bodyType::bodyType (
            const ::std::string & s,
            const ::xercesc::DOMElement * e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a string fragment.

Parameters

s	A string fragment to extract the data from.
e	A pointer to DOM element containing the string fragment.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.9.2.4 bodyType() [4/4]

```
Components::bodyType::bodyType (
            const bodyType & x,
```

```
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.9.3 Member Function Documentation

16.9.3.1 _clone()

```
bodyType * Components::bodyType::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

The documentation for this class was generated from the following files:

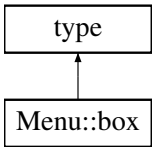
- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.10 Menu::box Class Reference

Class corresponding to the box schema type.

```
#include <menu.hxx>
```

Inheritance diagram for Menu::box:



position

Accessor and modifier functions for the position required element.

- typedef ::Common::position position_type

  Element type.
- typedef ::xsd::cxx::tree::traits< position_type, char > position_traits

  Element traits type.
- const position_type & position () const

  Return a read-only (constant) reference to the element.
- position_type & position ()

  Return a read-write reference to the element.
- void position (const position_type &x)

  Set the element value.
- void position (::std::unique_ptr< position_type > p)

  Set the element value without copying.

size

Accessor and modifier functions for the size required element.

- typedef ::Common::size size_type

  Element type.
- typedef ::xsd::cxx::tree::traits< size_type, char > size_traits

  Element traits type.
- const size_type & size () const

  Return a read-only (constant) reference to the element.
- size_type & size ()

  Return a read-write reference to the element.
- void size (const size_type &x)

  Set the element value.
- void size (::std::unique_ptr< size_type > p)

  Set the element value without copying.

color

Accessor and modifier functions for the color required element.

- typedef ::Common::color color_type

  Element type.
- typedef ::xsd::cxx::tree::traits< color_type, char > color_traits

  Element traits type.
- const color_type & color () const

  Return a read-only (constant) reference to the element.
- color_type & color ()

  Return a read-write reference to the element.
- void color (const color_type &x)

  Set the element value.
- void color (::std::unique_ptr< color_type > p)

  Set the element value without copying.

Constructors

- box (const position_type &, const size_type &, const color_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- box (::std::unique_ptr< position_type >, ::std::unique_ptr< size_type >, ::std::unique_ptr< color_type >)

  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- box (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- box (const box &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual box * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- box & operator= (const box &x)

  Copy assignment operator.
- virtual ~box ()

  Destructor.

16.10.1 Detailed Description

Class corresponding to the box schema type.

16.10.2 Constructor & Destructor Documentation

16.10.2.1 box() [1/3]

```
Menu::box::box (
            ::std::unique_ptr< position_type > position,
            ::std::unique_ptr< size_type > size,
            ::std::unique_ptr< color_type > color )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.10.2.2 box() [2/3]

```
Menu::box::box (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.10.2.3 box() [3/3]

```
Menu::box::box (
            const box & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.10.3 Member Function Documentation

16.10.3.1 _clone()

```
box * Menu::box::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.10.3.2 color() [1/4]

```
box::color_type & Menu::box::color ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.10.3.3 color() [2/4]

```
const box::color_type & Menu::box::color ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.10.3.4 color() [3/4]

```
void Menu::box::color (
            ::std::unique_ptr< color_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.10.3.5 color() [4/4]

```
void Menu::box::color (
            const color_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.10.3.6 operator=()

```
box & Menu::box::operator= (
            const box & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.10.3.7 position() [1/4]

```
box::position_type & Menu::box::position ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.10.3.8 position() [2/4]

```
const box::position_type & Menu::box::position ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.10.3.9 position() [3/4]

```
void Menu::box::position (
            ::std::unique_ptr< position_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.10.3.10 position() [4/4]

```
void Menu::box::position (
            const position_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.10.3.11 size() [1/4]

```
box::size_type & Menu::box::size ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.10.3.12 size() [2/4]

```
const box::size_type & Menu::box::size ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.10.3.13 size() [3/4]

```
void Menu::box::size (
            ::std::unique_ptr< size_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.10.3.14 size() [4/4]

```
void Menu::box::size (
            const size_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

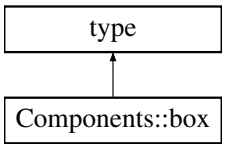
- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.11 Components::box Class Reference

Class corresponding to the box schema type.

`#include <components.hxx>`

Inheritance diagram for Components::box:



width

Accessor and modifier functions for the width required element.

- typedef ::xml_schema::float_ width_type
  Element type.
- typedef ::xsd::cxx::tree::traits< width_type, char > width_traits
  Element traits type.
- const width_type & width () const
  Return a read-only (constant) reference to the element.
- width_type & width ()
  Return a read-write reference to the element.
- void width (const width_type &x)
  Set the element value.

height

Accessor and modifier functions for the height required element.

- typedef ::xml_schema::float_ height_type
  Element type.
- typedef ::xsd::cxx::tree::traits< height_type, char > height_traits
  Element traits type.
- const height_type & height () const
  Return a read-only (constant) reference to the element.
- height_type & height ()
  Return a read-write reference to the element.
- void height (const height_type &x)
  Set the element value.

Constructors

- box (const width_type &, const height_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- box (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- box (const box &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual box * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- box & operator= (const box &x)
  Copy assignment operator.
- virtual ~box ()
  Destructor.

16.11.1 Detailed Description

Class corresponding to the box schema type.

16.11.2 Constructor & Destructor Documentation

16.11.2.1 box() [1/2]

```
Components::box::box (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.11.2.2 box() [2/2]

```
Components::box::box (
            const box & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.11.3 Member Function Documentation

16.11.3.1 _clone()

```
box * Components::box::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.11.3.2 height() [1/3]

```
box::height_type & Components::box::height ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.11.3.3 height() [2/3]

```
const box::height_type & Components::box::height ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.11.3.4 height() [3/3]

```
void Components::box::height (
            const height_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.11.3.5 operator=()

```
box & Components::box::operator= (
            const box & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.11.3.6 width() [1/3]

```
box::width_type & Components::box::width ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.11.3.7 width() [2/3]

```
const box::width_type & Components::box::width ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.11.3.8 width() [3/3]

```
void Components::box::width (
            const width_type & x )
```

Set the element value.

Parameters

x	A new value to set.

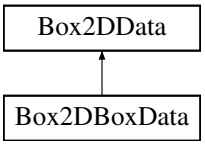
This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.12 Box2DBoxData Class Reference

Inheritance diagram for Box2DBoxData:



Public Attributes

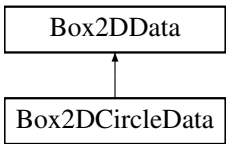
- Vector2 **size**

The documentation for this class was generated from the following file:

- Engine/Physics/PhysicsEngineAdapter.hpp

16.13 Box2DCircleData Class Reference

Inheritance diagram for Box2DCircleData:



Public Attributes

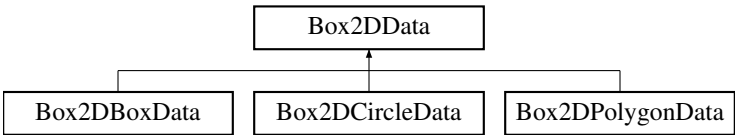
- float **radius**

The documentation for this class was generated from the following file:

- Engine/Physics/PhysicsEngineAdapter.hpp

16.14 Box2DData Class Reference

Inheritance diagram for Box2DData:



Public Attributes

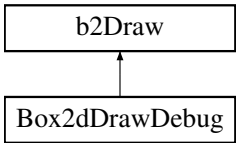
- BodyType **bodyType**
- Vector2 **position**
- bool **isBullet** = false
- bool **isSensor** = false
- bool **isEnabled** = true
- ContactHandler ∗ **userData** = nullptr

The documentation for this class was generated from the following file:

- Engine/Physics/PhysicsEngineAdapter.hpp

16.15 Box2dDrawDebug Class Reference

Inheritance diagram for Box2dDrawDebug:



Public Member Functions

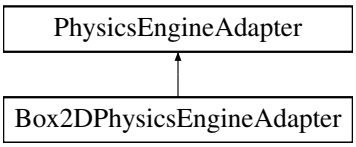
- **Box2dDrawDebug** (const SDLRenderingAdapter &adapter, SDL_Renderer &renderer)
- void **DrawPolygon** (const b2Vec2 ∗vertices, int32 vertexCount, const b2Color &color) override
- void **DrawSolidPolygon** (const b2Vec2 ∗vertices, int32 vertexCount, const b2Color &color) override
- void **DrawCircle** (const b2Vec2 &center, float radius, const b2Color &color) override
- void **DrawSolidCircle** (const b2Vec2 &center, float radius, const b2Vec2 &axis, const b2Color &color) override
- void **DrawSegment** (const b2Vec2 &p1, const b2Vec2 &p2, const b2Color &color) override
- void **DrawTransform** (const b2Transform &xf) override
- void **DrawPoint** (const b2Vec2 &p, float size, const b2Color &color) override

The documentation for this class was generated from the following file:

- Engine/Physics/PhysicsDebug/Box2dDrawDebug.hpp

16.16 Box2DPhysicsEngineAdapter Class Reference

Inheritance diagram for Box2DPhysicsEngineAdapter:



Public Member Functions

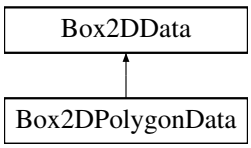
- void **update** (float deltaTime) override
- BodyId **createBody** (const Box2DBoxData &box2dBoxData) override
- BodyId **createBody** (const Box2DCircleData &box2DCircleData) override
- BodyId **createBody** (const Box2DPolygonData &box2DPolygonData) override
- void **referencePositionToBody** (BodyId bodyId, float &x, float &y) override
- RTransform **getRPosition** (BodyId bodyId) override
- void **destroyBody** (BodyId bodyID) override
- void **DebugDraw** (const SDLRenderingAdapter &renderingAdapter, SDL_Renderer &renderer) override
- void **getVelocity** (Vector2 &velocity, BodyId bodyId) const override
- void **addForce** (const BodyId i, Vector2 direction) const override
- void **setLinearVelocity** (BodyId bodyId, const Vector2 &vector2) override
- void **setTransform** (unsigned int bodyId, Vector2 pos, float angle) const override
- void **setFixedRotation** (BodyId bodyId, bool b) override
- void **setAngle** (BodyId bodyId, float angle) const override
- void **setEnabled** (BodyId id, bool b) const override
- bool **isWorldLocked** () const override

The documentation for this class was generated from the following files:

- Engine/Physics/Box2DPhysicsEngineAdapter.hpp
- Engine/Physics/Box2dPhysicsEngineAdapter.cpp

16.17 Box2DPolygonData Class Reference

Inheritance diagram for Box2DPolygonData:



Public Attributes

- std::vector< Vector2 > **points**

The documentation for this class was generated from the following file:

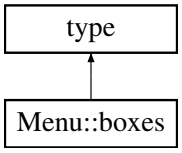
- Engine/Physics/PhysicsEngineAdapter.hpp

16.18 Menu::boxes Class Reference

Class corresponding to the boxes schema type.

`#include <menu.hxx>`

Inheritance diagram for Menu::boxes:



box

Accessor and modifier functions for the box sequence element.

- typedef ::Menu::box box_type
  Element type.
- typedef ::xsd::cxx::tree::sequence< box_type > box_sequence
  Element sequence container type.
- typedef box_sequence::iterator box_iterator
  Element iterator type.
- typedef box_sequence::const_iterator box_const_iterator
  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< box_type, char > box_traits
  Element traits type.
- const box_sequence & box () const
  Return a read-only (constant) reference to the element sequence.
- box_sequence & box ()
  Return a read-write reference to the element sequence.
- void box (const box_sequence &s)
  Copy elements from a given sequence.

Constructors

- boxes ()
  Create an instance from the ultimate base and initializers for required elements and attributes.
- boxes (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- boxes (const boxes &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual boxes * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- boxes & operator= (const boxes &x)
  Copy assignment operator.
- virtual ~boxes ()
  Destructor.

16.18.1 Detailed Description

Class corresponding to the boxes schema type.

16.18.2 Constructor & Destructor Documentation

16.18.2.1 boxes() [1/2]

```
Menu::boxes::boxes (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.18.2.2 boxes() [2/2]

```
Menu::boxes::boxes (
            const boxes & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.18.3 Member Function Documentation

16.18.3.1 _clone()

```
boxes * Menu::boxes::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.18.3.2 box() [1/3]

```
boxes::box_sequence & Menu::boxes::box ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.18.3.3 box() [2/3]

```
const boxes::box_sequence & Menu::boxes::box ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.18.3.4 box() [3/3]

```
void Menu::boxes::box (
            const box_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

16.18.3.5 operator=()

```
boxes & Menu::boxes::operator= (
            const boxes & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

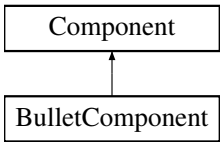
For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.19 BulletComponent Class Reference

Inheritance diagram for BulletComponent:



Public Member Functions

- **BulletComponent** (EntityId id)
- void **render** () override
- void **update** (const Input &inputSystem) override
- void **fixedUpdate** (const float &deltaTime) override
- std::string **name** () const override
- Component * **build** (EntityId entityId, const Components::component *component) override
- void **initialize** (EntityObject &entityParent) override

Additional Inherited Members

The documentation for this class was generated from the following files:

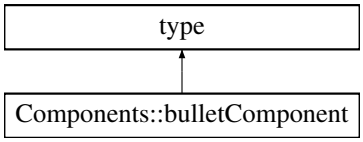
- Game/Components/BulletComponent.hpp
- Game/Components/BulletComponent.cpp

16.20 Components::bulletComponent Class Reference

Class corresponding to the bulletComponent schema type.

`#include <components.hxx>`

Inheritance diagram for Components::bulletComponent:



Constructors

- bulletComponent ()
  Create an instance from the ultimate base and initializers for required elements and attributes.
- bulletComponent (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- bulletComponent (const ::xercesc::DOMAttr &a, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM attribute.
- bulletComponent (const ::std::string &s, const ::xercesc::DOMElement *e, ::xml_schema::flags f=0, ↩ ::xml_schema::container *c=0)
  Create an instance from a string fragment.
- bulletComponent (const bulletComponent &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual bulletComponent * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- virtual ~bulletComponent ()
  Destructor.

16.20.1 Detailed Description

Class corresponding to the bulletComponent schema type.

16.20.2 Constructor & Destructor Documentation

16.20.2.1 bulletComponent() [1/4]

```
Components::bulletComponent::bulletComponent (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.20.2.2 bulletComponent() [2/4]

```
Components::bulletComponent::bulletComponent (
            const ::xercesc::DOMAttr & a,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM attribute.

Parameters

a	A DOM attribute to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.20.2.3 bulletComponent() [3/4]

```
Components::bulletComponent::bulletComponent (
            const ::std::string & s,
            const ::xercesc::DOMElement * e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a string fragment.

Parameters

s	A string fragment to extract the data from.
e	A pointer to DOM element containing the string fragment.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.20.2.4 bulletComponent() [4/4]

```
Components::bulletComponent::bulletComponent (
            const bulletComponent & x,
```

```
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.20.3 Member Function Documentation

16.20.3.1 _clone()

```
bulletComponent * Components::bulletComponent::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

The documentation for this class was generated from the following files:

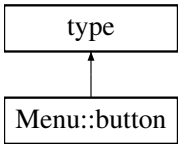
- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.21 Menu::button Class Reference

Class corresponding to the button schema type.

```
#include <menu.hxx>
```

Inheritance diagram for Menu::button:



position

Accessor and modifier functions for the position required element.

- typedef ::Common::position position_type

  Element type.
- typedef ::xsd::cxx::tree::traits< position_type, char > position_traits

  Element traits type.
- const position_type & position () const

  Return a read-only (constant) reference to the element.
- position_type & position ()

  Return a read-write reference to the element.
- void position (const position_type &x)

  Set the element value.
- void position (::std::unique_ptr< position_type > p)

  Set the element value without copying.

size

Accessor and modifier functions for the size required element.

- typedef ::Common::size size_type

  Element type.
- typedef ::xsd::cxx::tree::traits< size_type, char > size_traits

  Element traits type.
- const size_type & size () const

  Return a read-only (constant) reference to the element.
- size_type & size ()

  Return a read-write reference to the element.
- void size (const size_type &x)

  Set the element value.
- void size (::std::unique_ptr< size_type > p)

  Set the element value without copying.

text

Accessor and modifier functions for the text optional element.

- typedef ::Menu::text1 text_type

  Element type.
- typedef ::xsd::cxx::tree::optional< text_type > text_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< text_type, char > text_traits

  Element traits type.
- const text_optional & text () const

  Return a read-only (constant) reference to the element container.
- text_optional & text ()

  Return a read-write reference to the element container.
- void text (const text_type &x)

  Set the element value.
- void text (const text_optional &x)

  Set the element value.
- void text (::std::unique_ptr< text_type > p)

  Set the element value without copying.

color

Accessor and modifier functions for the color optional element.

- typedef ::Common::color color_type

  Element type.
- typedef ::xsd::cxx::tree::optional< color_type > color_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< color_type, char > color_traits

  Element traits type.
- const color_optional & color () const

  Return a read-only (constant) reference to the element container.
- color_optional & color ()

  Return a read-write reference to the element container.
- void color (const color_type &x)

  Set the element value.
- void color (const color_optional &x)

  Set the element value.
- void color (::std::unique_ptr< color_type > p)

  Set the element value without copying.

resources

Accessor and modifier functions for the resources optional element.

- typedef ::Menu::resources resources_type
  Element type.
- typedef ::xsd::cxx::tree::optional< resources_type > resources_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< resources_type, char > resources_traits
  Element traits type.
- const resources_optional & resources () const
  Return a read-only (constant) reference to the element container.
- resources_optional & resources ()
  Return a read-write reference to the element container.
- void resources (const resources_type &x)
  Set the element value.
- void resources (const resources_optional &x)
  Set the element value.
- void resources (::std::unique_ptr< resources_type > p)
  Set the element value without copying.

events

Accessor and modifier functions for the events required element.

- typedef ::Common::events events_type
  Element type.
- typedef ::xsd::cxx::tree::traits< events_type, char > events_traits
  Element traits type.
- const events_type & events () const
  Return a read-only (constant) reference to the element.
- events_type & events ()
  Return a read-write reference to the element.
- void events (const events_type &x)
  Set the element value.
- void events (::std::unique_ptr< events_type > p)
  Set the element value without copying.

Constructors

- button (const position_type &, const size_type &, const events_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- button (::std::unique_ptr< position_type >, ::std::unique_ptr< size_type >, ::std::unique_ptr< events_type >)
  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- button (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.

- button (const button &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual button * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- button & operator= (const button &x)

  Copy assignment operator.
- virtual ~button ()

  Destructor.

16.21.1 Detailed Description

Class corresponding to the button schema type.

16.21.2 Constructor & Destructor Documentation

16.21.2.1 button() [1/3]

```
Menu::button::button (
            ::std::unique_ptr< position_type > position,
            ::std::unique_ptr< size_type > size,
            ::std::unique_ptr< events_type > events )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.21.2.2 button() [2/3]

```
Menu::button::button (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.21.2.3 button() [3/3]

```
Menu::button::button (
            const button & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.21.3 Member Function Documentation

16.21.3.1 _clone()

```
button * Menu::button::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.21.3.2 color() [1/5]

```
button::color_optional & Menu::button::color ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.21.3.3 color() [2/5]

```
const button::color_optional & Menu::button::color ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.21.3.4 color() [3/5]

```
void Menu::button::color (
            ::std::unique_ptr< color_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.21.3.5 color() [4/5]

```
void Menu::button::color (
            const color_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.21.3.6 color() [5/5]

```
void Menu::button::color (
            const color_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.21.3.7 events() [1/4]

```
button::events_type & Menu::button::events ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.21.3.8 events() [2/4]

```
const button::events_type & Menu::button::events ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.21.3.9 events() [3/4]

```
void Menu::button::events (
            ::std::unique_ptr< events_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.21.3.10 events() [4/4]

```
void Menu::button::events (
            const events_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.21.3.11 operator=()

```
button & Menu::button::operator= (
            const button & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.21.3.12 position() [1/4]

```
button::position_type & Menu::button::position ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.21.3.13 position() [2/4]

```
const button::position_type & Menu::button::position ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.21.3.14 position() [3/4]

```
void Menu::button::position (
            ::std::unique_ptr< position_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.21.3.15 position() [4/4]

```
void Menu::button::position (
            const position_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.21.3.16 resources() [1/5]

```
button::resources_optional & Menu::button::resources ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.21.3.17 resources() [2/5]

```
const button::resources_optional & Menu::button::resources ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.21.3.18 resources() [3/5]

```
void Menu::button::resources (
            ::std::unique_ptr< resources_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.21.3.19 resources() [4/5]

```
void Menu::button::resources (
            const resources_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.21.3.20 resources() [5/5]

```
void Menu::button::resources (
            const resources_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.21.3.21 size() [1/4]

```
button::size_type & Menu::button::size ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.21.3.22 size() [2/4]

```
const button::size_type & Menu::button::size ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.21.3.23 size() [3/4]

```
void Menu::button::size (
            ::std::unique_ptr< size_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.21.3.24 size() [4/4]

```
void Menu::button::size (
            const size_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.21.3.25 text() [1/5]

```
button::text_optional & Menu::button::text ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.21.3.26 text() [2/5]

```
const button::text_optional & Menu::button::text ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.21.3.27 text() [3/5]

```
void Menu::button::text (
            ::std::unique_ptr< text_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.21.3.28 text() [4/5]

```
void Menu::button::text (
            const text_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.21.3.29 text() [5/5]

```
void Menu::button::text (
            const text_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

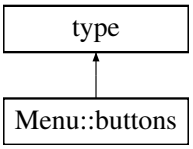
- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.22 Menu::buttons Class Reference

Class corresponding to the buttons schema type.

`#include <menu.hxx>`

Inheritance diagram for Menu::buttons:



button

Accessor and modifier functions for the button sequence element.

- typedef ::Menu::button button_type

  Element type.
- typedef ::xsd::cxx::tree::sequence< button_type > button_sequence

  Element sequence container type.
- typedef button_sequence::iterator button_iterator

  Element iterator type.
- typedef button_sequence::const_iterator button_const_iterator

  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< button_type, char > button_traits

  Element traits type.
- const button_sequence & button () const

  Return a read-only (constant) reference to the element sequence.
- button_sequence & button ()

  Return a read-write reference to the element sequence.
- void button (const button_sequence &s)

  Copy elements from a given sequence.

Constructors

- buttons ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- buttons (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- buttons (const buttons &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual buttons * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- buttons & operator= (const buttons &x)

  Copy assignment operator.
- virtual ~buttons ()

  Destructor.

16.22.1 Detailed Description

Class corresponding to the buttons schema type.

16.22.2 Constructor & Destructor Documentation

16.22.2.1 buttons() [1/2]

```
Menu::buttons::buttons (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.22.2.2 buttons() [2/2]

```
Menu::buttons::buttons (
            const buttons & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.22.3 Member Function Documentation

16.22.3.1 _clone()

```
buttons * Menu::buttons::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.22.3.2 button() [1/3]

```
buttons::button_sequence & Menu::buttons::button ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.22.3.3 button() [2/3]

```
const buttons::button_sequence & Menu::buttons::button ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.22.3.4 button() [3/3]

```
void Menu::buttons::button (
            const button_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

16.22.3.5 operator=()

```
buttons & Menu::buttons::operator= (
            const buttons & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

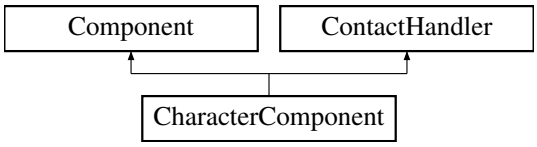
For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.23 CharacterComponent Class Reference

Inheritance diagram for CharacterComponent:



Public Member Functions

- **CharacterComponent** (EntityId id)
- **CharacterComponent** (EntityId id, const Vector2 &position)
- void **getVelocity** (Vector2 &velocity)
- void **setVelocity** (const Vector2 &velocity)
- float **getHealth** ()
- void **setHealth** (float hp)
- void **die** ()
- void **doDamage** (float hp)
- void **fixedUpdate** (const float &deltaTime) override
- const Spritesheet & **getSpriteSheet** ()
- Component ∗ **build** (EntityId entityId, const Components::component ∗component) override
- void **initialize** (EntityObject &entityParent) override
- std::string **name** () const override
- void **startContact** (b2Contact ∗contact) override
- void **endContact** (b2Contact ∗contact) override
- void **render** () override
- void **update** (const Input &inputSystem) override

Additional Inherited Members

The documentation for this class was generated from the following files:

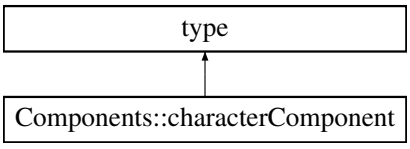
- Game/Components/CharacterComponent.hpp
- Game/Components/CharacterComponent.cpp

16.24 Components::characterComponent Class Reference

Class corresponding to the characterComponent schema type.

`#include <components.hxx>`

Inheritance diagram for Components::characterComponent:



Constructors

- characterComponent ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- characterComponent (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- characterComponent (const ::xercesc::DOMAttr &a, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM attribute.
- characterComponent (const ::std::string &s, const ::xercesc::DOMElement *e, ::xml_schema::flags f=0, ↩ ::xml_schema::container *c=0)

  Create an instance from a string fragment.
- characterComponent (const characterComponent &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual characterComponent * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- virtual ~characterComponent ()

  Destructor.

16.24.1 Detailed Description

Class corresponding to the characterComponent schema type.

16.24.2 Constructor & Destructor Documentation

16.24.2.1 characterComponent() [1/4]

```
Components::characterComponent::characterComponent (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.24.2.2 characterComponent() [2/4]

```
Components::characterComponent::characterComponent (
```

```
            const ::xercesc::DOMAttr & a,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM attribute.

Parameters

a	A DOM attribute to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.24.2.3 characterComponent() [3/4]

```
Components::characterComponent::characterComponent (
            const ::std::string & s,
            const ::xercesc::DOMElement * e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a string fragment.

Parameters

s	A string fragment to extract the data from.
e	A pointer to DOM element containing the string fragment.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.24.2.4 characterComponent() [4/4]

```
Components::characterComponent::characterComponent (
            const characterComponent & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.24.3 Member Function Documentation

16.24.3.1 _clone()

```
characterComponent * Components::characterComponent::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

The documentation for this class was generated from the following files:

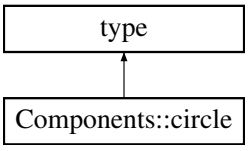
- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.25 Components::circle Class Reference

Class corresponding to the circle schema type.

```
#include <components.hxx>
```

Inheritance diagram for Components::circle:



position

Accessor and modifier functions for the position required element.

- typedef ::Common::position position_type
  Element type.
- typedef ::xsd::cxx::tree::traits< position_type, char > position_traits
  Element traits type.
- const position_type & position () const
  Return a read-only (constant) reference to the element.
- position_type & position ()
  Return a read-write reference to the element.
- void position (const position_type &x)
  Set the element value.
- void position (::std::unique_ptr< position_type > p)
  Set the element value without copying.

radius

Accessor and modifier functions for the radius required element.

- typedef ::xml_schema::float_ radius_type
  Element type.
- typedef ::xsd::cxx::tree::traits< radius_type, char > radius_traits
  Element traits type.
- const radius_type & radius () const
  Return a read-only (constant) reference to the element.
- radius_type & radius ()
  Return a read-write reference to the element.
- void radius (const radius_type &x)
  Set the element value.

Constructors

- circle (const position_type &, const radius_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- circle (::std::unique_ptr< position_type >, const radius_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- circle (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- circle (const circle &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual circle * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- circle & operator= (const circle &x)
  Copy assignment operator.
- virtual ~circle ()
  Destructor.

16.25.1 Detailed Description

Class corresponding to the circle schema type.

16.25.2 Constructor & Destructor Documentation

16.25.2.1 circle() [1/3]

```
Components::circle::circle (
            ::std::unique_ptr< position_type > position,
            const radius_type & radius )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.25.2.2 circle() [2/3]

```
Components::circle::circle (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.25.2.3 circle() [3/3]

```
Components::circle::circle (
            const circle & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the _clone function instead.

16.25.3 Member Function Documentation

16.25.3.1 _clone()

```
circle * Components::circle::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.25.3.2 operator=()

```
circle & Components::circle::operator= (
            const circle & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the _clone function instead.

16.25.3.3 position() [1/4]

```
circle::position_type & Components::circle::position ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.25.3.4 position() [2/4]

```
const circle::position_type & Components::circle::position ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.25.3.5 position() [3/4]

```
void Components::circle::position (
            ::std::unique_ptr< position_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.25.3.6 position() [4/4]

```
void Components::circle::position (
            const position_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.25.3.7 radius() [1/3]

```
circle::radius_type & Components::circle::radius ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.25.3.8 radius() [2/3]

```
const circle::radius_type & Components::circle::radius ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.25.3.9 radius() [3/3]

```
void Components::circle::radius (
            const radius_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

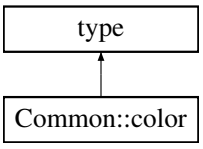
- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.26 Common::color Class Reference

Class corresponding to the color schema type.

```
#include <common.hxx>
```

Inheritance diagram for Common::color:



hex

Accessor and modifier functions for the hex required element.

- typedef ::xml_schema::string hex_type
  Element type.
- typedef ::xsd::cxx::tree::traits< hex_type, char > hex_traits
  Element traits type.
- const hex_type & hex () const
  Return a read-only (constant) reference to the element.
- hex_type & hex ()
  Return a read-write reference to the element.
- void hex (const hex_type &x)
  Set the element value.
- void hex (::std::unique_ptr< hex_type > p)
  Set the element value without copying.

alpha

Accessor and modifier functions for the alpha required element.

- typedef ::Common::alpha alpha_type
  Element type.
- typedef ::xsd::cxx::tree::traits< alpha_type, char > alpha_traits
  Element traits type.
- const alpha_type & alpha () const
  Return a read-only (constant) reference to the element.
- alpha_type & alpha ()
  Return a read-write reference to the element.
- void alpha (const alpha_type &x)
  Set the element value.
- void alpha (::std::unique_ptr< alpha_type > p)
  Set the element value without copying.

Constructors

- color (const hex_type &, const alpha_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- color (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- color (const color &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual color * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- color & operator= (const color &x)
  Copy assignment operator.
- virtual ~color ()
  Destructor.

16.26.1 Detailed Description

Class corresponding to the color schema type.

16.26.2 Constructor & Destructor Documentation

16.26.2.1 color() [1/2]

```
Common::color::color (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.26.2.2 color() [2/2]

```
Common::color::color (
            const color & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.26.3 Member Function Documentation

16.26.3.1 _clone()

```
color * Common::color::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.26.3.2 alpha() [1/4]

```
color::alpha_type & Common::color::alpha ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.26.3.3 alpha() [2/4]

```
const color::alpha_type & Common::color::alpha ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.26.3.4 alpha() [3/4]

```
void Common::color::alpha (
            ::std::unique_ptr< alpha_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.26.3.5 alpha() [4/4]

```
void Common::color::alpha (
            const alpha_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.26.3.6 hex() [1/4]

```
color::hex_type & Common::color::hex ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.26.3.7 hex() [2/4]

```
const color::hex_type & Common::color::hex ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.26.3.8 hex() [3/4]

```
void Common::color::hex (
            ::std::unique_ptr< hex_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.26.3.9 hex() [4/4]

```
void Common::color::hex (
            const hex_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.26.3.10 operator=()

```
color & Common::color::operator= (
            const color & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

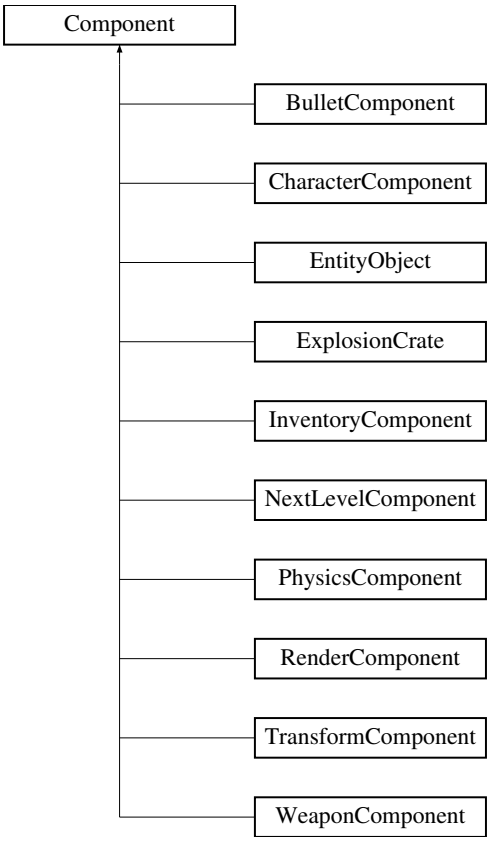
For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.27 Component Class Reference

Inheritance diagram for Component:



Public Member Functions

- **Component** (EntityId id, Component &component)
- const EntityId **getEntityId** ()
- virtual void **render** ()=0
- virtual void **update** (const Input &inputSystem)=0
- virtual void **fixedUpdate** (const float &deltaTime)=0
- **Component** (EntityId id)
- virtual std::string **name** () const =0
- virtual Component * **build** (EntityId entityId, const Components::component *component)=0
- virtual void **initialize** (EntityObject &entityParent)=0

Protected Attributes

- EntityId **owner**

The documentation for this class was generated from the following file:

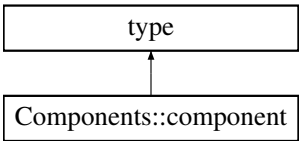
- Game/Components/Component.hpp

16.28 Components::component Class Reference

Class corresponding to the component schema type.

`#include <components.hxx>`

Inheritance diagram for Components::component:



componentName

Accessor and modifier functions for the componentName required element.

- typedef ::Components::componentName componentName_type
  Element type.
- typedef ::xsd::cxx::tree::traits< componentName_type, char > componentName_traits
  Element traits type.
- const componentName_type & componentName () const
  Return a read-only (constant) reference to the element.
- componentName_type & componentName ()
  Return a read-write reference to the element.
- void componentName (const componentName_type &x)
  Set the element value.
- void componentName (::std::unique_ptr< componentName_type > p)
  Set the element value without copying.

transformComponent

Accessor and modifier functions for the transformComponent optional element.

- typedef ::Components::transformComponent transformComponent_type
  Element type.
- typedef ::xsd::cxx::tree::optional< transformComponent_type > transformComponent_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< transformComponent_type, char > transformComponent_traits
  Element traits type.
- const transformComponent_optional & transformComponent () const
  Return a read-only (constant) reference to the element container.
- transformComponent_optional & transformComponent ()
  Return a read-write reference to the element container.
- void transformComponent (const transformComponent_type &x)
  Set the element value.
- void transformComponent (const transformComponent_optional &x)
  Set the element value.
- void transformComponent (::std::unique_ptr< transformComponent_type > p)
  Set the element value without copying.

renderComponent

Accessor and modifier functions for the renderComponent optional element.

- typedef ::Components::renderComponent renderComponent_type

  Element type.
- typedef ::xsd::cxx::tree::optional< renderComponent_type > renderComponent_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< renderComponent_type, char > renderComponent_traits

  Element traits type.
- const renderComponent_optional & renderComponent () const

  Return a read-only (constant) reference to the element container.
- renderComponent_optional & renderComponent ()

  Return a read-write reference to the element container.
- void renderComponent (const renderComponent_type &x)

  Set the element value.
- void renderComponent (const renderComponent_optional &x)

  Set the element value.
- void renderComponent (::std::unique_ptr< renderComponent_type > p)

  Set the element value without copying.

physicsComponent

Accessor and modifier functions for the physicsComponent optional element.

- typedef ::Components::physicsComponent physicsComponent_type

  Element type.
- typedef ::xsd::cxx::tree::optional< physicsComponent_type > physicsComponent_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< physicsComponent_type, char > physicsComponent_traits

  Element traits type.
- const physicsComponent_optional & physicsComponent () const

  Return a read-only (constant) reference to the element container.
- physicsComponent_optional & physicsComponent ()

  Return a read-write reference to the element container.
- void physicsComponent (const physicsComponent_type &x)

  Set the element value.
- void physicsComponent (const physicsComponent_optional &x)

  Set the element value.
- void physicsComponent (::std::unique_ptr< physicsComponent_type > p)

  Set the element value without copying.

characterComponent

Accessor and modifier functions for the characterComponent optional element.

- typedef ::Components::characterComponent characterComponent_type

  Element type.
- typedef ::xsd::cxx::tree::optional< characterComponent_type > characterComponent_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< characterComponent_type, char > characterComponent_traits

  Element traits type.
- const characterComponent_optional & characterComponent () const

  Return a read-only (constant) reference to the element container.
- characterComponent_optional & characterComponent ()

  Return a read-write reference to the element container.
- void characterComponent (const characterComponent_type &x)

  Set the element value.
- void characterComponent (const characterComponent_optional &x)

  Set the element value.
- void characterComponent (::std::unique_ptr< characterComponent_type > p)

  Set the element value without copying.

explosionCrate

Accessor and modifier functions for the explosionCrate optional element.

- typedef ::Components::explosionCrate explosionCrate_type

  Element type.
- typedef ::xsd::cxx::tree::optional< explosionCrate_type > explosionCrate_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< explosionCrate_type, char > explosionCrate_traits

  Element traits type.
- const explosionCrate_optional & explosionCrate () const

  Return a read-only (constant) reference to the element container.
- explosionCrate_optional & explosionCrate ()

  Return a read-write reference to the element container.
- void explosionCrate (const explosionCrate_type &x)

  Set the element value.
- void explosionCrate (const explosionCrate_optional &x)

  Set the element value.
- void explosionCrate (::std::unique_ptr< explosionCrate_type > p)

  Set the element value without copying.

bulletComponent

Accessor and modifier functions for the bulletComponent optional element.

- typedef ::Components::bulletComponent bulletComponent_type

  Element type.
- typedef ::xsd::cxx::tree::optional< bulletComponent_type > bulletComponent_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< bulletComponent_type, char > bulletComponent_traits

  Element traits type.
- const bulletComponent_optional & bulletComponent () const

  Return a read-only (constant) reference to the element container.
- bulletComponent_optional & bulletComponent ()

  Return a read-write reference to the element container.
- void bulletComponent (const bulletComponent_type &x)

  Set the element value.
- void bulletComponent (const bulletComponent_optional &x)

  Set the element value.
- void bulletComponent (::std::unique_ptr< bulletComponent_type > p)

  Set the element value without copying.

nextLevelComponent

Accessor and modifier functions for the nextLevelComponent optional element.

- typedef ::Components::nextLevelComponent nextLevelComponent_type

  Element type.
- typedef ::xsd::cxx::tree::optional< nextLevelComponent_type > nextLevelComponent_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< nextLevelComponent_type, char > nextLevelComponent_traits

  Element traits type.
- const nextLevelComponent_optional & nextLevelComponent () const

  Return a read-only (constant) reference to the element container.
- nextLevelComponent_optional & nextLevelComponent ()

  Return a read-write reference to the element container.
- void nextLevelComponent (const nextLevelComponent_type &x)

  Set the element value.
- void nextLevelComponent (const nextLevelComponent_optional &x)

  Set the element value.
- void nextLevelComponent (::std::unique_ptr< nextLevelComponent_type > p)

  Set the element value without copying.

Constructors

- component (const componentName_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- component (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- component (const component &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual component * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- component & operator= (const component &x)

  Copy assignment operator.
- virtual ~component ()

  Destructor.

16.28.1 Detailed Description

Class corresponding to the component schema type.

16.28.2 Constructor & Destructor Documentation

16.28.2.1 component() [1/2]

```
Components::component::component (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.28.2.2 component() [2/2]

```
Components::component::component (
            const component & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.28.3 Member Function Documentation

16.28.3.1 _clone()

```
component * Components::component::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.28.3.2 bulletComponent() [1/5]

```
component::bulletComponent_optional & Components::component::bulletComponent ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.28.3.3 bulletComponent() [2/5]

```
const component::bulletComponent_optional & Components::component::bulletComponent ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.28.3.4 bulletComponent() [3/5]

```
void Components::component::bulletComponent (
            ::std::unique_ptr< bulletComponent_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.28.3.5 bulletComponent() [4/5]

```
void Components::component::bulletComponent (
            const bulletComponent_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.28.3.6 bulletComponent() [5/5]

```
void Components::component::bulletComponent (
            const bulletComponent_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.28.3.7 characterComponent() [1/5]

```
component::characterComponent_optional & Components::component::characterComponent ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.28.3.8 characterComponent() [2/5]

```
const component::characterComponent_optional & Components::component::characterComponent ( )
const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.28.3.9 characterComponent() [3/5]

```
void Components::component::characterComponent (
            ::std::unique_ptr< characterComponent_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.28.3.10 characterComponent() [4/5]

```
void Components::component::characterComponent (
            const characterComponent_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.28.3.11 characterComponent() [5/5]

```
void Components::component::characterComponent (
            const characterComponent_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.28.3.12 componentName() [1/4]

```
component::componentName_type & Components::component::componentName ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.28.3.13 componentName() [2/4]

```
const component::componentName_type & Components::component::componentName ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.28.3.14 componentName() [3/4]

```
void Components::component::componentName (
            ::std::unique_ptr< componentName_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.28.3.15 componentName() [4/4]

```
void Components::component::componentName (
            const componentName_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.28.3.16 explosionCrate() [1/5]

```
component::explosionCrate_optional & Components::component::explosionCrate ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.28.3.17 explosionCrate() [2/5]

```
const component::explosionCrate_optional & Components::component::explosionCrate ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.28.3.18 explosionCrate() [3/5]

```
void Components::component::explosionCrate (
            ::std::unique_ptr< explosionCrate_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.28.3.19 explosionCrate() [4/5]

```
void Components::component::explosionCrate (
            const explosionCrate_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.28.3.20 explosionCrate() [5/5]

```
void Components::component::explosionCrate (
            const explosionCrate_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.28.3.21 nextLevelComponent() [1/5]

```
component::nextLevelComponent_optional & Components::component::nextLevelComponent ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.28.3.22 nextLevelComponent() [2/5]

```
const component::nextLevelComponent_optional & Components::component::nextLevelComponent ( )
const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.28.3.23 nextLevelComponent() [3/5]

```
void Components::component::nextLevelComponent (
            ::std::unique_ptr< nextLevelComponent_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.28.3.24 nextLevelComponent() [4/5]

```
void Components::component::nextLevelComponent (
            const nextLevelComponent_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.28.3.25 nextLevelComponent() [5/5]

```
void Components::component::nextLevelComponent (
            const nextLevelComponent_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.28.3.26 operator=()

```
component & Components::component::operator= (
            const component & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.28.3.27 physicsComponent() [1/5]

```
component::physicsComponent_optional & Components::component::physicsComponent ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.28.3.28 physicsComponent() [2/5]

```
const component::physicsComponent_optional & Components::component::physicsComponent ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.28.3.29 physicsComponent() [3/5]

```
void Components::component::physicsComponent (
            ::std::unique_ptr< physicsComponent_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.28.3.30 physicsComponent() [4/5]

```
void Components::component::physicsComponent (
            const physicsComponent_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.28.3.31 physicsComponent() [5/5]

```
void Components::component::physicsComponent (
            const physicsComponent_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.28.3.32 renderComponent() [1/5]

```
component::renderComponent_optional & Components::component::renderComponent ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.28.3.33 renderComponent() [2/5]

```
const component::renderComponent_optional & Components::component::renderComponent ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.28.3.34 renderComponent() [3/5]

```
void Components::component::renderComponent (
            ::std::unique_ptr< renderComponent_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.28.3.35 renderComponent() [4/5]

```
void Components::component::renderComponent (
            const renderComponent_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.28.3.36 renderComponent() [5/5]

```
void Components::component::renderComponent (
            const renderComponent_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.28.3.37 transformComponent() [1/5]

```
component::transformComponent_optional & Components::component::transformComponent ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.28.3.38 transformComponent() [2/5]

```
const component::transformComponent_optional & Components::component::transformComponent ( )
const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.28.3.39 transformComponent() [3/5]

```
void Components::component::transformComponent (
            ::std::unique_ptr< transformComponent_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.28.3.40 transformComponent() [4/5]

```
void Components::component::transformComponent (
            const transformComponent_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.28.3.41 transformComponent() [5/5]

```
void Components::component::transformComponent (
            const transformComponent_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.29 ComponentFactory Class Reference

Public Member Functions

- template<class T >
  T * **getComponent** (const EntityId &id)
- Component * **getComponent** (const EntityId &id, const std::string &name, const Components::component *loadedComponent)

Static Public Member Functions

- static bool **IsPhysicsComponent** (const std::string &componentName)

The documentation for this class was generated from the following files:

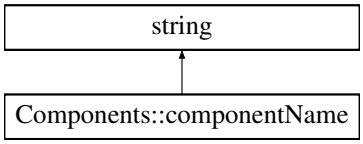
- Game/Components/ComponentFactory.hpp
- Game/Components/ComponentFactory.cpp

16.30 Components::componentName Class Reference

Class corresponding to the componentName schema type.

`#include <components.hxx>`

Inheritance diagram for Components::componentName:



Constructors

- componentName ()
  Create an instance from initializers for required elements and attributes.
- componentName (const char ∗)
  Create an instance from a C string and initializers for required elements and attributes.
- componentName (const ::std::string &)
  Create an instance from a string andinitializers for required elements and attributes.
- componentName (const ::xml_schema::string &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- componentName (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container ∗c=0)
  Create an instance from a DOM element.
- componentName (const ::xercesc::DOMAttr &a, ::xml_schema::flags f=0, ::xml_schema::container ∗c=0)
  Create an instance from a DOM attribute.
- componentName (const ::std::string &s, const ::xercesc::DOMElement ∗e, ::xml_schema::flags f=0, ↩ ::xml_schema::container ∗c=0)
  Create an instance from a string fragment.
- componentName (const componentName &x, ::xml_schema::flags f=0, ::xml_schema::container ∗c=0)
  Copy constructor.
- virtual componentName ∗ _clone (::xml_schema::flags f=0, ::xml_schema::container ∗c=0) const
  Copy the instance polymorphically.
- virtual ∼componentName ()
  Destructor.

16.30.1 Detailed Description

Class corresponding to the componentName schema type.

16.30.2 Constructor & Destructor Documentation

16.30.2.1 componentName() [1/4]

```
Components::componentName::componentName (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.30.2.2 componentName() [2/4]

```
Components::componentName::componentName (
            const ::xercesc::DOMAttr & a,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM attribute.

Parameters

a	A DOM attribute to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.30.2.3 componentName() [3/4]

```
Components::componentName::componentName (
            const ::std::string & s,
            const ::xercesc::DOMElement * e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a string fragment.

Parameters

s	A string fragment to extract the data from.
e	A pointer to DOM element containing the string fragment.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.30.2.4 componentName() [4/4]

```
Components::componentName::componentName (
            const componentName & x,
```

```
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.30.3 Member Function Documentation

16.30.3.1 _clone()

```
componentName * Components::componentName::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

The documentation for this class was generated from the following files:

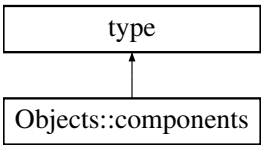
- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.31 Objects::components Class Reference

Class corresponding to the components schema type.

```
#include <objects.hxx>
```

Inheritance diagram for Objects::components:



component

Accessor and modifier functions for the component sequence element.

- typedef ::Components::component component_type

  Element type.
- typedef ::xsd::cxx::tree::sequence< component_type > component_sequence

  Element sequence container type.
- typedef component_sequence::iterator component_iterator

  Element iterator type.
- typedef component_sequence::const_iterator component_const_iterator

  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< component_type, char > component_traits

  Element traits type.
- const component_sequence & component () const

  Return a read-only (constant) reference to the element sequence.
- component_sequence & component ()

  Return a read-write reference to the element sequence.
- void component (const component_sequence &s)

  Copy elements from a given sequence.

Constructors

- components ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- components (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- components (const components &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual components * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- components & operator= (const components &x)

  Copy assignment operator.
- virtual ~components ()

  Destructor.

16.31.1 Detailed Description

Class corresponding to the components schema type.

16.31.2 Constructor & Destructor Documentation

16.31.2.1 components() [1/2]

```
Objects::components::components (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.31.2.2 components() [2/2]

```
Objects::components::components (
            const components & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.31.3 Member Function Documentation

16.31.3.1 _clone()

```
components * Objects::components::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.31.3.2 component() [1/3]

```
components::component_sequence & Objects::components::component ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.31.3.3 component() [2/3]

```
const components::component_sequence & Objects::components::component ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.31.3.4 component() [3/3]

```
void Objects::components::component (
            const component_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

16.31.3.5 operator=()

```
components & Objects::components::operator= (
             const components & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

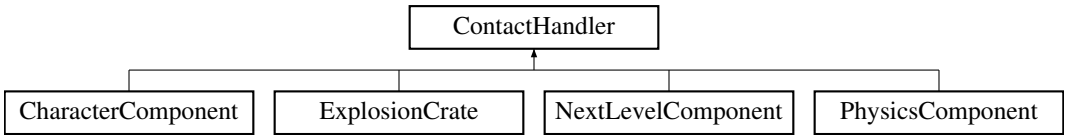
For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/objects.hxx
- Resources/XML/Generated/objects.cxx

16.32 ContactHandler Class Reference

Inheritance diagram for ContactHandler:



Public Member Functions

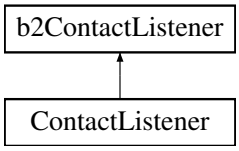
- virtual void **startContact** (b2Contact ∗contact)=0
- virtual void **endContact** (b2Contact ∗contact)=0

The documentation for this class was generated from the following file:

- Engine/Physics/ContactHandler.hpp

16.33 ContactListener Class Reference

Inheritance diagram for ContactListener:



The documentation for this class was generated from the following file:

- Engine/Physics/ContactListener.hpp

16.34 Engine Class Reference

Public Member Functions

- **Engine** (Engine &other)=delete
- void **operator=** (const Engine &)=delete
- SDL_Renderer * **getRenderer** ()
- TextureManager * **getTextureManager** ()

Static Public Member Functions

- static Engine * **getInstance** ()
- static void initWindow (int SCREEN_WIDTH, int SCREEN_HEIGHT)
- static void closeWindow ()

16.34.1 Member Function Documentation

16.34.1.1 closeWindow()

```
void Engine::closeWindow ( )  [static]
```

Close the game window

Clears the SDL_Window pointer in the Engine class and calls SDL_DestroyWindow & SDL_Quit.

16.34.1.2 initWindow()

```
void Engine::initWindow (
            int SCREEN_WIDTH,
            int SCREEN_HEIGHT )  [static]
```

Initialize the game window

Stores a SDL_Window pointer in the Engine class.

Parameters

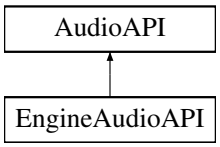
SCREEN_WIDTH	The width of the window.
SCREEN_HEIGHT	The height of the window.

The documentation for this class was generated from the following files:

- Engine/Engine.hpp
- Engine/Engine.cpp

16.35 EngineAudioAPI Class Reference

Inheritance diagram for EngineAudioAPI:



Public Member Functions

- EngineAudioAPI ()
- **EngineAudioAPI** (const EngineAudioAPI &other)=default
- **EngineAudioAPI** (EngineAudioAPI &&other) noexcept=default
- EngineAudioAPI & **operator=** (const EngineAudioAPI &other)=default
- EngineAudioAPI & **operator=** (EngineAudioAPI &&other) noexcept=default
- std::vector< std::string > **getAudioNames** () override
- void **playFromMemory** (const std::string &name) override
- void **playFromPath** (const std::string &path, AudioType &type) override
- void **loadInMemory** (const std::string &path, const std::string &name, AudioType type) override
- void **stopAudio** () override
- void **stopMusic** () override
- void **stopSound** (int channel) override
- void **stopSounds** () override
- void **changeMasterVolume** (int volume) override
- void **changeChannelVolume** (int channel, int volume) override
- void **changeMusicVolume** (int volume) override
- int **getChannelsAverageVolume** () override
- int **getChannelVolume** (int channel) override
- int **getMusicVolume** () override
- void **toggleMusic** () override
- void **toggleSound** (int channel) override
- void **toggleSounds** () override

16.35.1 Constructor & Destructor Documentation

16.35.1.1 EngineAudioAPI()

```
EngineAudioAPI::EngineAudioAPI ( )
```

A class that separates the adapter and the Game layer

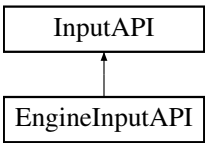
The documentation for this class was generated from the following files:

- API/Audio/EngineAudioAPI.hpp
- API/Audio/EngineAudioAPI.cpp

16.36 EngineInputAPI Class Reference

```
#include <EngineInputAPI.hpp>
```

Inheritance diagram for EngineInputAPI:



Public Member Functions

- Input **getInput** () const override
- Event< Input > & **getInputEvent** () override
- void **getMousePosition** (int &x, int &y) const override

16.36.1 Detailed Description

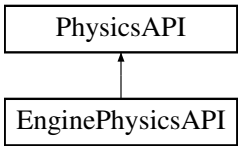
RAII class for managing it's adapter

The documentation for this class was generated from the following files:

- API/Input/EngineInputAPI.hpp
- API/Input/EngineInputAPI.cpp

16.37 EnginePhysicsAPI Class Reference

Inheritance diagram for EnginePhysicsAPI:



Public Member Functions

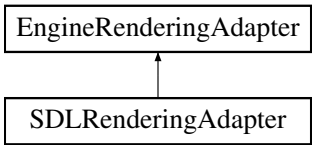
- void **update** (float timeStep) override
- BodyId **createBody** (const Box2DBoxData &box2DBoxData) const override
- BodyId **createBody** (const Box2DCircleData &box2DCircleData) const override
- BodyId **createBody** (const Box2DPolygonData &box2DPolygonData) const override
- void **destroyBody** (BodyId bodyID) override
- RTransform **getRPosition** (BodyId bodyId) const override
- void **DebugDraw** (const RenderingAPI &renderingApi, SDL_Renderer &renderer) override
- void **GetVelocity** (Vector2 &velocity, const BodyId bodyId) const override
- void **setLinearVelocity** (const BodyId bodyId, const Vector2 &vector2) const override
- void **setFixedRotation** (const BodyId i, bool b) const override
- void **addForce** (const BodyId i, Vector2 direction) const override
- void **setAngle** (const BodyId i, float angle) const override
- void **destroyBody** (BodyId i) const override
- void **setTransform** (unsigned int bodyId, Vector2 pos, float angle) const override
- void **setEnabled** (BodyId id, bool b) const override
- PhysicsEngineAdapter & **getPhysicsEngineAdapter** () const override

The documentation for this class was generated from the following file:

- API/Physics/EnginePhysicsAPI.hpp

16.38 EngineRenderingAdapter Class Reference

Inheritance diagram for EngineRenderingAdapter:



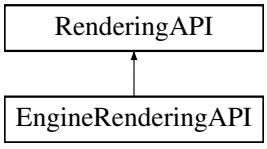
The documentation for this class was generated from the following file:

- Engine/Rendering/Adapter/EngineRenderingAdapter.h

16.39 EngineRenderingAPI Class Reference

```
#include <EngineRenderingAPI.hpp>
```

Inheritance diagram for EngineRenderingAPI:



Public Member Functions

- void drawTexture (const std::string &textureId, float x, float y, float width, float height, double scale, double r) const override
- Spritesheet * createSpriteSheet (std::string path, std::string spriteSheetId, int width, int height) const override
- Spritesheet * **createSpriteSheet** (const std::string &path, const std::string &jsonPath, std::string &sprite↩ SheetId)
- void **createText** (const std::string &fontName, const std::string &text, int fontSize, const std::string &hex, const std::string &textureId) const override
- void **drawLine** (Vector2 &a, Vector2 &b) const override
- bool loadTexture (const std::string &path, const std::string &textureId) override
- void **drawRectangle** (Vector2 &position, float width, float height, std::string &color, float opacity) const override
- void **drawBackground** (std::string &hex, float alpha) const override
- TMXLevel * **loadTMX** (const LevelData &levelData, PhysicsEngineAdapter &physicsEngineAdapter) override
- void **render** () const override

Static Public Member Functions

- static TextureManager * **GetTextureManager** ()

16.39.1 Detailed Description

This class acts as a facade for the _engine, it stores and references variables and constants needed for rendering and resource allocation like textures.

16.39.2 Member Function Documentation

16.39.2.1 createSpriteSheet()

```
Spritesheet * EngineRenderingAPI::createSpriteSheet (
            std::string path,
            std::string spriteSheetId,
            int width,
            int height ) const  [override], [virtual]
```

Parameters

path	
sprite↩ SheetId	
rows	
columns	
width	
height	

Returns

Spritesheet

Implements RenderingAPI.

16.39.2.2 drawTexture()

```
void EngineRenderingAPI::drawTexture (
            const std::string & textureId,
            float x,
            float y,
            float width,
            float height,
            double scale,
            double r ) const  [override], [virtual]
```

Parameters

textureId	
x	
y	
width	
height	
scale	
r	

Implements RenderingAPI.

16.39.2.3 loadTexture()

```
bool EngineRenderingAPI::loadTexture (
            const std::string & path,
            const std::string & textureId )  [override], [virtual]
```

Parameters

path	
textureId	

Returns

success

Implements RenderingAPI.

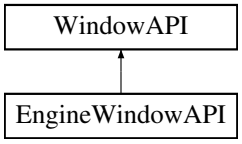
The documentation for this class was generated from the following files:

- API/Rendering/EngineRenderingAPI.hpp
- API/Rendering/EngineRenderingAPI.cpp

16.40 EngineWindowAPI Class Reference

`#include <EngineWindowAPI.hpp>`

Inheritance diagram for EngineWindowAPI:



Public Member Functions

- void **initWindow** (int SCREEN_WIDTH, int SCREEN_HEIGHT) const override
- void **closeWindow** () const override
- SDL_Renderer * **getRenderer** () const override

Public Attributes

- Engine & **_engine**

16.40.1 Detailed Description

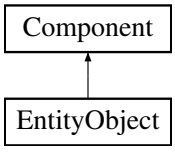
RAII class for managing the _engine

The documentation for this class was generated from the following files:

- API/Engine/EngineWindowAPI.hpp
- API/Engine/EngineWindowAPI.cpp

16.41 EntityObject Class Reference

Inheritance diagram for EntityObject:



Public Member Functions

- **EntityObject** (EntityId id, std::string name="")
- TransformComponent * **getTransform** ()
- PhysicsComponent * **getPhysicsComponent** ()
- const std::vector< std::unique_ptr< Component > > & **getComponents** ()
- template<class T >
  T * **getComponent** () const
- Component * **getComponent** (std::string componentName)
- void **initializeComponents** ()
- void **render** () override
- void **update** (const Input &inputSystem) override
- void **fixedUpdate** (const float &deltaTime) override
- void **addComponent** (Component *component)
- std::string **name** () const override
- Component * **build** (EntityId entityId, const Components::component *component) override
- void **initialize** (EntityObject &entityParent) override

Public Attributes

- std::string **entityName**

Additional Inherited Members

The documentation for this class was generated from the following files:

- Game/Components/EntityObject.hpp
- Game/Components/EntityObject.cpp

16.42 EntityXMLParser Class Reference

Static Public Member Functions

- static void **createEntities** (std::multimap< std::string, Components::component * > &outEntities, xsd::cxx↩
  ::tree::sequence< Objects::object > &objects)

The documentation for this class was generated from the following files:

- Engine/XMLParser/EntityXMLParser.hpp
- Engine/XMLParser/EntityXMLParser.cpp

16.43 Event< tArg0 > Class Template Reference

Public Member Functions

- Event & **operator+=** (funcPtr function)
- void **operator()** (tArg0 arg)

The documentation for this class was generated from the following file:

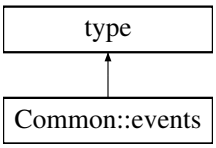
- API/Helpers/Event.h

16.44 Common::events Class Reference

Class corresponding to the events schema type.

`#include <common.hxx>`

Inheritance diagram for Common::events:



onEnter

Accessor and modifier functions for the onEnter optional element.

- typedef ::Common::onEnter onEnter_type
  Element type.
- typedef ::xsd::cxx::tree::optional< onEnter_type > onEnter_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< onEnter_type, char > onEnter_traits
  Element traits type.
- const onEnter_optional & onEnter () const
  Return a read-only (constant) reference to the element container.
- onEnter_optional & onEnter ()
  Return a read-write reference to the element container.
- void onEnter (const onEnter_type &x)
  Set the element value.
- void onEnter (const onEnter_optional &x)
  Set the element value.
- void onEnter (::std::unique_ptr< onEnter_type > p)
  Set the element value without copying.

onLeave

Accessor and modifier functions for the onLeave optional element.

- typedef ::Common::onLeave onLeave_type
  Element type.
- typedef ::xsd::cxx::tree::optional< onLeave_type > onLeave_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< onLeave_type, char > onLeave_traits
  Element traits type.
- const onLeave_optional & onLeave () const
  Return a read-only (constant) reference to the element container.
- onLeave_optional & onLeave ()
  Return a read-write reference to the element container.
- void onLeave (const onLeave_type &x)
  Set the element value.
- void onLeave (const onLeave_optional &x)
  Set the element value.
- void onLeave (::std::unique_ptr< onLeave_type > p)
  Set the element value without copying.

onAttacked

Accessor and modifier functions for the onAttacked optional element.

- typedef ::Common::onAttacked onAttacked_type

  Element type.
- typedef ::xsd::cxx::tree::optional< onAttacked_type > onAttacked_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< onAttacked_type, char > onAttacked_traits

  Element traits type.
- const onAttacked_optional & onAttacked () const

  Return a read-only (constant) reference to the element container.
- onAttacked_optional & onAttacked ()

  Return a read-write reference to the element container.
- void onAttacked (const onAttacked_type &x)

  Set the element value.
- void onAttacked (const onAttacked_optional &x)

  Set the element value.
- void onAttacked (::std::unique_ptr< onAttacked_type > p)

  Set the element value without copying.

onDestroyed

Accessor and modifier functions for the onDestroyed optional element.

- typedef ::Common::onDestroyed onDestroyed_type

  Element type.
- typedef ::xsd::cxx::tree::optional< onDestroyed_type > onDestroyed_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< onDestroyed_type, char > onDestroyed_traits

  Element traits type.
- const onDestroyed_optional & onDestroyed () const

  Return a read-only (constant) reference to the element container.
- onDestroyed_optional & onDestroyed ()

  Return a read-write reference to the element container.
- void onDestroyed (const onDestroyed_type &x)

  Set the element value.
- void onDestroyed (const onDestroyed_optional &x)

  Set the element value.
- void onDestroyed (::std::unique_ptr< onDestroyed_type > p)

  Set the element value without copying.

onAttack

Accessor and modifier functions for the onAttack optional element.

- typedef ::Common::onAttack onAttack_type

  Element type.
- typedef ::xsd::cxx::tree::optional< onAttack_type > onAttack_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< onAttack_type, char > onAttack_traits

  Element traits type.
- const onAttack_optional & onAttack () const

  Return a read-only (constant) reference to the element container.
- onAttack_optional & onAttack ()

  Return a read-write reference to the element container.
- void onAttack (const onAttack_type &x)

  Set the element value.
- void onAttack (const onAttack_optional &x)

  Set the element value.
- void onAttack (::std::unique_ptr< onAttack_type > p)

  Set the element value without copying.

onClick

Accessor and modifier functions for the onClick optional element.

- typedef ::Common::onClick onClick_type

  Element type.
- typedef ::xsd::cxx::tree::optional< onClick_type > onClick_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< onClick_type, char > onClick_traits

  Element traits type.
- const onClick_optional & onClick () const

  Return a read-only (constant) reference to the element container.
- onClick_optional & onClick ()

  Return a read-write reference to the element container.
- void onClick (const onClick_type &x)

  Set the element value.
- void onClick (const onClick_optional &x)

  Set the element value.
- void onClick (::std::unique_ptr< onClick_type > p)

  Set the element value without copying.

Constructors

- events ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- events (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- events (const events &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual events * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- events & operator= (const events &x)

  Copy assignment operator.
- virtual ~events ()

  Destructor.

16.44.1 Detailed Description

Class corresponding to the events schema type.

16.44.2 Constructor & Destructor Documentation

16.44.2.1 events() [1/2]

```
Common::events::events (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.44.2.2 events() [2/2]

```
Common::events::events (
            const events & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.44.3 Member Function Documentation

16.44.3.1 _clone()

```
events * Common::events::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.44.3.2 onAttack() [1/5]

```
events::onAttack_optional & Common::events::onAttack ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.44.3.3 onAttack() [2/5]

```
const events::onAttack_optional & Common::events::onAttack ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.44.3.4 onAttack() [3/5]

```
void Common::events::onAttack (
            ::std::unique_ptr< onAttack_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.44.3.5 onAttack() [4/5]

```
void Common::events::onAttack (
            const onAttack_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.44.3.6 onAttack() [5/5]

```
void Common::events::onAttack (
            const onAttack_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.44.3.7 onAttacked() [1/5]

```
events::onAttacked_optional & Common::events::onAttacked ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.44.3.8 onAttacked() [2/5]

```
const events::onAttacked_optional & Common::events::onAttacked ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.44.3.9 onAttacked() [3/5]

```
void Common::events::onAttacked (
            ::std::unique_ptr< onAttacked_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.44.3.10 onAttacked() [4/5]

```
void Common::events::onAttacked (
            const onAttacked_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.44.3.11 onAttacked() [5/5]

```
void Common::events::onAttacked (
            const onAttacked_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.44.3.12 onClick() [1/5]

```
events::onClick_optional & Common::events::onClick ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.44.3.13 onClick() [2/5]

```
const events::onClick_optional & Common::events::onClick ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.44.3.14 onClick() [3/5]

```
void Common::events::onClick (
            ::std::unique_ptr< onClick_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.44.3.15 onClick() [4/5]

```
void Common::events::onClick (
            const onClick_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.44.3.16 onClick() [5/5]

```
void Common::events::onClick (
            const onClick_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.44.3.17 onDestroyed() [1/5]

```
events::onDestroyed_optional & Common::events::onDestroyed ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.44.3.18 onDestroyed() [2/5]

```
const events::onDestroyed_optional & Common::events::onDestroyed ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.44.3.19 onDestroyed() [3/5]

```
void Common::events::onDestroyed (
            ::std::unique_ptr< onDestroyed_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.44.3.20 onDestroyed() [4/5]

```
void Common::events::onDestroyed (
            const onDestroyed_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.44.3.21 onDestroyed() [5/5]

```
void Common::events::onDestroyed (
            const onDestroyed_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.44.3.22 onEnter() [1/5]

```
events::onEnter_optional & Common::events::onEnter ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.44.3.23 onEnter() [2/5]

```
const events::onEnter_optional & Common::events::onEnter ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.44.3.24 onEnter() [3/5]

```
void Common::events::onEnter (
            ::std::unique_ptr< onEnter_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.44.3.25 onEnter() [4/5]

```
void Common::events::onEnter (
            const onEnter_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.44.3.26 onEnter() [5/5]

```
void Common::events::onEnter (
            const onEnter_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.44.3.27 onLeave() [1/5]

```
events::onLeave_optional & Common::events::onLeave ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.44.3.28 onLeave() [2/5]

```
const events::onLeave_optional & Common::events::onLeave ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.44.3.29 onLeave() [3/5]

```
void Common::events::onLeave (
            ::std::unique_ptr< onLeave_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.44.3.30 onLeave() [4/5]

```
void Common::events::onLeave (
            const onLeave_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.44.3.31 onLeave() [5/5]

```
void Common::events::onLeave (
            const onLeave_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.44.3.32 operator=()

```
events & Common::events::operator= (
            const events & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

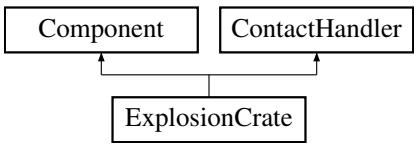
For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.45 ExplosionCrate Class Reference

Inheritance diagram for ExplosionCrate:



Public Member Functions

- **ExplosionCrate** (EntityId id)
- void **render** () override
- void **update** (const Input &inputSystem) override
- void **fixedUpdate** (const float &deltaTime) override
- std::string **name** () const override
- Component * **build** (EntityId entityId, const Components::component *component) override
- void **initialize** (EntityObject &entityParent) override
- void **startContact** (b2Contact *contact) override
- void **endContact** (b2Contact *contact) override

Additional Inherited Members

The documentation for this class was generated from the following file:

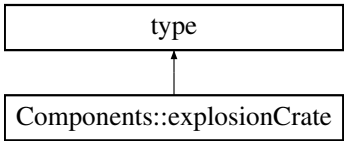
- Game/ContactHandlers/ExplosionCrate.hpp

16.46 Components::explosionCrate Class Reference

Class corresponding to the explosionCrate schema type.

`#include <components.hxx>`

Inheritance diagram for Components::explosionCrate:



Constructors

- explosionCrate ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- explosionCrate (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- explosionCrate (const ::xercesc::DOMAttr &a, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM attribute.
- explosionCrate (const ::std::string &s, const ::xercesc::DOMElement *e, ::xml_schema::flags f=0, ↩ ::xml_schema::container *c=0)

  Create an instance from a string fragment.
- explosionCrate (const explosionCrate &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual explosionCrate * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- virtual ~explosionCrate ()

  Destructor.

16.46.1 Detailed Description

Class corresponding to the explosionCrate schema type.

16.46.2 Constructor & Destructor Documentation

16.46.2.1 explosionCrate() [1/4]

```
Components::explosionCrate::explosionCrate (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.46.2.2 explosionCrate() [2/4]

```
Components::explosionCrate::explosionCrate (
            const ::xercesc::DOMAttr & a,
```

```
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM attribute.

Parameters

a	A DOM attribute to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.46.2.3 explosionCrate() [3/4]

```
Components::explosionCrate::explosionCrate (
            const ::std::string & s,
            const ::xercesc::DOMElement * e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a string fragment.

Parameters

s	A string fragment to extract the data from.
e	A pointer to DOM element containing the string fragment.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.46.2.4 explosionCrate() [4/4]

```
Components::explosionCrate::explosionCrate (
            const explosionCrate & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.46.3 Member Function Documentation

16.46.3.1 _clone()

```
explosionCrate * Components::explosionCrate::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

The documentation for this class was generated from the following files:

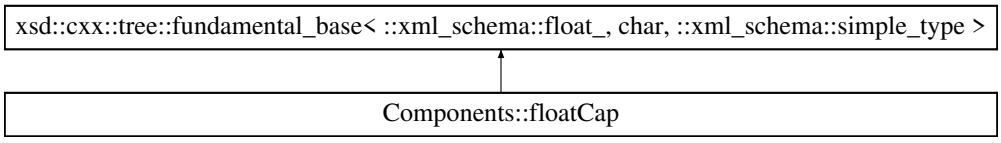
- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.47 Components::floatCap Class Reference

Class corresponding to the floatCap schema type.

```
#include <components.hxx>
```

Inheritance diagram for Components::floatCap:



Constructors

- floatCap (const ::xml_schema::float_ &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- floatCap (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- floatCap (const ::xercesc::DOMAttr &a, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM attribute.
- floatCap (const ::std::string &s, const ::xercesc::DOMElement *e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a string fragment.
- floatCap (const floatCap &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual floatCap * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- virtual ~floatCap ()

  Destructor.

16.47.1 Detailed Description

Class corresponding to the floatCap schema type.

16.47.2 Constructor & Destructor Documentation

16.47.2.1 floatCap() [1/4]

```
Components::floatCap::floatCap (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.47.2.2 floatCap() [2/4]

```
Components::floatCap::floatCap (
            const ::xercesc::DOMAttr & a,
```

```
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM attribute.

Parameters

a	A DOM attribute to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.47.2.3 floatCap() [3/4]

```
Components::floatCap::floatCap (
            const ::std::string & s,
            const ::xercesc::DOMElement * e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a string fragment.

Parameters

s	A string fragment to extract the data from.
e	A pointer to DOM element containing the string fragment.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.47.2.4 floatCap() [4/4]

```
Components::floatCap::floatCap (
            const floatCap & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.47.3 Member Function Documentation

16.47.3.1 _clone()

```
floatCap * Components::floatCap::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

The documentation for this class was generated from the following files:

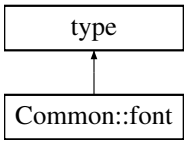
- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.48 Common::font Class Reference

Class corresponding to the font schema type.

```
#include <common.hxx>
```

Inheritance diagram for Common::font:



family

Accessor and modifier functions for the family required element.

- typedef ::xml_schema::string family_type
  Element type.
- typedef ::xsd::cxx::tree::traits< family_type, char > family_traits
  Element traits type.
- const family_type & family () const
  Return a read-only (constant) reference to the element.
- family_type & family ()
  Return a read-write reference to the element.
- void family (const family_type &x)
  Set the element value.
- void family (::std::unique_ptr< family_type > p)
  Set the element value without copying.

weight

Accessor and modifier functions for the weight required element.

- typedef ::xml_schema::string weight_type
  Element type.
- typedef ::xsd::cxx::tree::traits< weight_type, char > weight_traits
  Element traits type.
- const weight_type & weight () const
  Return a read-only (constant) reference to the element.
- weight_type & weight ()
  Return a read-write reference to the element.
- void weight (const weight_type &x)
  Set the element value.
- void weight (::std::unique_ptr< weight_type > p)
  Set the element value without copying.

size

Accessor and modifier functions for the size required element.

- typedef ::xml_schema::int_ size_type
  Element type.
- typedef ::xsd::cxx::tree::traits< size_type, char > size_traits
  Element traits type.
- const size_type & size () const
  Return a read-only (constant) reference to the element.
- size_type & size ()
  Return a read-write reference to the element.
- void size (const size_type &x)
  Set the element value.

Constructors

- font (const family_type &, const weight_type &, const size_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- font (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- font (const font &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual font * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- font & operator= (const font &x)

  Copy assignment operator.
- virtual ~font ()

  Destructor.

16.48.1 Detailed Description

Class corresponding to the font schema type.

16.48.2 Constructor & Destructor Documentation

16.48.2.1 font() [1/2]

```
Common::font::font (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.48.2.2 font() [2/2]

```
Common::font::font (
            const font & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.48.3 Member Function Documentation

16.48.3.1 _clone()

```
font * Common::font::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.48.3.2 family() [1/4]

```
font::family_type & Common::font::family ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.48.3.3 family() [2/4]

```
const font::family_type & Common::font::family ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.48.3.4 family() [3/4]

```
void Common::font::family (
            ::std::unique_ptr< family_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.48.3.5 family() [4/4]

```
void Common::font::family (
            const family_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.48.3.6 operator=()

```
font & Common::font::operator= (
            const font & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.48.3.7 size() [1/3]

```
font::size_type & Common::font::size ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.48.3.8 size() [2/3]

```
const font::size_type & Common::font::size ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.48.3.9 size() [3/3]

```
void Common::font::size (
            const size_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.48.3.10 weight() [1/4]

```
font::weight_type & Common::font::weight ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.48.3.11 weight() [2/4]

```
const font::weight_type & Common::font::weight ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.48.3.12 weight() [3/4]

```
void Common::font::weight (
            ::std::unique_ptr< weight_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.48.3.13 weight() [4/4]

```
void Common::font::weight (
            const weight_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.49 FrameCounter Class Reference

Public Member Functions

- **FrameCounter** (RenderingAPI &_renderingAPI)
- void **render** ()

The documentation for this class was generated from the following files:

- Game/UI/FrameCounter.h
- Game/UI/FrameCounter.cpp

16.50 Game Class Reference

Public Member Functions

- **Game** (Game &other)=delete
- void **operator=** (const Game &)=delete
- void **initialize** ()
- void gameLoop ()
- PoolLevel * **getPoolLevel** ()
- EntityId createEntity ()
- void addComponent (EntityId id, Component *comp)
- template<typename T >
  T * getComponent (EntityId id)
- System< Component > getComponents (EntityId id)
- template<typename T >
  System< T > getComponents (EntityId id)
- InputAPI & **getInputAPI** ()
- PhysicsAPI & **getPhysicsAPI** ()
- RenderingAPI & **getRenderingApi** ()
- ComponentFactory * **getComponentFactory** ()
- void **initializeLeveL** (const std::string &levelName, const LevelData &data)
- void **addEventBodyHandler** (const std::function< void()> &function)
- void **unloadLevel** ()
- void **FixedUpdate** (float deltaTime)

Static Public Member Functions

- static Game * getInstance ()

16.50.1 Member Function Documentation

16.50.1.1 addComponent()

```
void Game::addComponent (
            EntityId id,
            Component * comp )
```

Add a component to the specified entity.

Parameters

id	
comp	

16.50.1.2 createEntity()

```
EntityId Game::createEntity ( )
```

Creates an entity and pushes it to the entity collection.

Returns

EntityId id - The id of the newly created entity.

16.50.1.3 gameLoop()

```
void Game::gameLoop ( )
```

Gameloop

16.50.1.4 getComponent()

```
template<typename T >
T * Game::getComponent (
             EntityId id )
```

Gets a single component of specified type.

Template Parameters

T	

Parameters

id	

Returns

16.50.1.5 getComponents() [1/2]

```
System< Component > Game::getComponents (
            EntityId id )
```

Gets components by entity id of all types.

Parameters

id	

Returns

16.50.1.6 getComponents() [2/2]

```
template<typename T >
System< T > Game::getComponents (
            EntityId id )
```

Gets components by entity id of a specified type.

Template Parameters

T	

Parameters

id	

Returns

16.50.1.7 getInstance()

```
Game * Game::getInstance ( )  [static]
```

The first time we call getInstance we will lock the storage location and then we make sure again that the variable is null and then we set the value.

The documentation for this class was generated from the following files:

- Game/Game.hpp
- Game/Game.cpp

16.51 GameTime Class Reference

Public Member Functions

- void **update** ()
- float **getTime** () const
- float **getDeltaTime** () const
- float **getTotalTime** () const
- float **getAccumulator** () const
- float **getFrameTimeSeconds** () const
- float **getFrameTimeMiliSeconds** () const
- void **resetTotalTime** ()
- void **setAccumulator** (float value)
- void **setDeltaTime** (float value)
- void **setTime** (float value)
- Event< float > & **getFixedUpdateEvent** ()

Static Public Member Functions

- static GameTime & **getInstance** ()

The documentation for this class was generated from the following files:

- Game/Helpers/GameTime.h
- Game/Helpers/GameTime.cpp

16.52 GlobalObjects Class Reference

Public Member Functions

- void **initializeObjects** (const std::string &name, const std::string &path, const std::string &poolName, const std::string &poolPath)
- std::unique_ptr< EntityObject > **loadEntity** (const std::string &fromList, const std::string &entityName)
- void **loadEntities** (std::vector< std::unique_ptr< EntityObject >> &entities, const std::string &fromList, const std::string &entityName, int amount)

Static Public Member Functions

- static GlobalObjects * **getInstance** ()

The documentation for this class was generated from the following files:

- Game/Object/GlobalObjects.hpp
- Game/Object/GlobalObjects.cpp

16.53 HealthComponent Class Reference

Public Member Functions

- float **getHealth** () const
- void **setHealth** (float newHealthPoints)
- void **doDamage** (float amountOfHealthPoints)
- void **die** ()

The documentation for this class was generated from the following files:

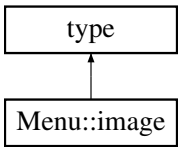
- Game/Components/HealthComponent.hpp
- Game/Components/HealthComponent.cpp

16.54 Menu::image Class Reference

Class corresponding to the image schema type.

`#include <menu.hxx>`

Inheritance diagram for Menu::image:



position

Accessor and modifier functions for the position required element.

- typedef ::Common::position position_type

  Element type.
- typedef ::xsd::cxx::tree::traits< position_type, char > position_traits

  Element traits type.
- const position_type & position () const

  Return a read-only (constant) reference to the element.
- position_type & position ()

  Return a read-write reference to the element.
- void position (const position_type &x)

  Set the element value.
- void position (::std::unique_ptr< position_type > p)

  Set the element value without copying.

size

Accessor and modifier functions for the size required element.

- typedef ::Common::size size_type
  Element type.
- typedef ::xsd::cxx::tree::traits< size_type, char > size_traits
  Element traits type.
- const size_type & size () const
  Return a read-only (constant) reference to the element.
- size_type & size ()
  Return a read-write reference to the element.
- void size (const size_type &x)
  Set the element value.
- void size (::std::unique_ptr< size_type > p)
  Set the element value without copying.

resources

Accessor and modifier functions for the resources required element.

- typedef ::Common::resources resources_type
  Element type.
- typedef ::xsd::cxx::tree::traits< resources_type, char > resources_traits
  Element traits type.
- const resources_type & resources () const
  Return a read-only (constant) reference to the element.
- resources_type & resources ()
  Return a read-write reference to the element.
- void resources (const resources_type &x)
  Set the element value.
- void resources (::std::unique_ptr< resources_type > p)
  Set the element value without copying.

Constructors

- image (const position_type &, const size_type &, const resources_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- image (::std::unique_ptr< position_type >, ::std::unique_ptr< size_type >, ::std::unique_ptr< resources_type >)
  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- image (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- image (const image &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual image * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- image & operator= (const image &x)
  Copy assignment operator.
- virtual ~image ()
  Destructor.

16.54.1 Detailed Description

Class corresponding to the image schema type.

16.54.2 Constructor & Destructor Documentation

16.54.2.1 image() [1/3]

```
Menu::image::image (
            ::std::unique_ptr< position_type > position,
            ::std::unique_ptr< size_type > size,
            ::std::unique_ptr< resources_type > resources )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.54.2.2 image() [2/3]

```
Menu::image::image (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.54.2.3 image() [3/3]

```
Menu::image::image (
            const image & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.54.3 Member Function Documentation

16.54.3.1 _clone()

```
image * Menu::image::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.54.3.2 operator=()

```
image & Menu::image::operator= (
            const image & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.54.3.3 position() [1/4]

```
image::position_type & Menu::image::position ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.54.3.4 position() [2/4]

```
const image::position_type & Menu::image::position ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.54.3.5 position() [3/4]

```
void Menu::image::position (
            ::std::unique_ptr< position_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.54.3.6 position() [4/4]

```
void Menu::image::position (
            const position_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.54.3.7 resources() [1/4]

```
image::resources_type & Menu::image::resources ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.54.3.8 resources() [2/4]

```
const image::resources_type & Menu::image::resources ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.54.3.9 resources() [3/4]

```
void Menu::image::resources (
            ::std::unique_ptr< resources_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.54.3.10 resources() [4/4]

```
void Menu::image::resources (
            const resources_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.54.3.11 size() [1/4]

```
image::size_type & Menu::image::size ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.54.3.12 size() [2/4]

```
const image::size_type & Menu::image::size ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.54.3.13 size() [3/4]

```
void Menu::image::size (
            ::std::unique_ptr< size_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.54.3.14 size() [4/4]

```
void Menu::image::size (
            const size_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

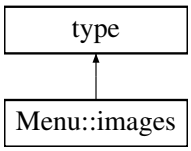
- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.55 Menu::images Class Reference

Class corresponding to the images schema type.

```
#include <menu.hxx>
```

Inheritance diagram for Menu::images:



image

Accessor and modifier functions for the image sequence element.

- typedef ::Menu::image image_type

  Element type.
- typedef ::xsd::cxx::tree::sequence< image_type > image_sequence

  Element sequence container type.
- typedef image_sequence::iterator image_iterator

  Element iterator type.
- typedef image_sequence::const_iterator image_const_iterator

  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< image_type, char > image_traits

  Element traits type.
- const image_sequence & image () const

  Return a read-only (constant) reference to the element sequence.
- image_sequence & image ()

  Return a read-write reference to the element sequence.
- void image (const image_sequence &s)

  Copy elements from a given sequence.

Constructors

- images ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- images (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- images (const images &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual images * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- images & operator= (const images &x)

  Copy assignment operator.
- virtual ~images ()

  Destructor.

16.55.1 Detailed Description

Class corresponding to the images schema type.

16.55.2 Constructor & Destructor Documentation

16.55.2.1 images() [1/2]

```
Menu::images::images (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.55.2.2 images() [2/2]

```
Menu::images::images (
            const images & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.55.3 Member Function Documentation

16.55.3.1 _clone()

```
images * Menu::images::_clone (
             ::xml_schema::flags f = 0,
             ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.55.3.2 image() [1/3]

```
images::image_sequence & Menu::images::image ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.55.3.3 image() [2/3]

```
const images::image_sequence & Menu::images::image ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.55.3.4 image() [3/3]

```
void Menu::images::image (
            const image_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

16.55.3.5 operator=()

```
images & Menu::images::operator= (
            const images & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.56 Input Struct Reference

`#include <Input.hpp>`

Public Types

- enum device {
  NONE = -1, KEYBOARD = 0, MOUSE = 1, CONTROLLER = 2,
  OTHER = 3 }

Public Attributes

- enum Input::device **device**
- float x
- float y
- InputAction keyMap

16.56.1 Detailed Description

A structure to represent an input from the Engine.

16.56.2 Member Enumeration Documentation

16.56.2.1 device

`enum Input::device`

Device the input was received from

Enumerator

NONE	Device -1: No Input
KEYBOARD	Device 0: Keyboard
MOUSE	Device 1: Mouse
CONTROLLER	Device 2: Controller
OTHER	Device 3: Other (e.g. the console)

16.56.3 Member Data Documentation

16.56.3.1 keyMap

```
InputAction Input::keyMap
```

InputAction struct containing code and action

16.56.3.2 x

```
float Input::x
```

X coordinate within the window

16.56.3.3 y

```
float Input::y
```

Y coordinate within the window

The documentation for this struct was generated from the following file:

- Engine/Input/Input.hpp

16.57 InputAction Struct Reference

```
#include <Input.hpp>
```

Public Attributes

- std::string code
- std::string action
- uint32_t **type**

16.57.1 Detailed Description

A structure to represent an input from the user.

16.57.2 Member Data Documentation

16.57.2.1 action

```
std::string InputAction::action
```

Action that is mapped to the keycode

16.57.2.2 code

```
std::string InputAction::code
```

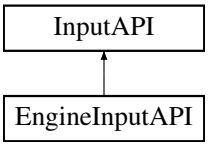
Keycode returned form the Keymap

The documentation for this struct was generated from the following file:

- Engine/Input/Input.hpp

16.58 InputAPI Class Reference

Inheritance diagram for InputAPI:



Public Member Functions

- virtual Input **getInput** () const =0
- virtual Event< Input > & **getInputEvent** ()=0
- virtual void **getMousePosition** (int &x, int &y) const =0

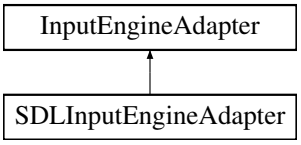
The documentation for this class was generated from the following file:

- API/Input/InputAPI.hpp

16.59 InputEngineAdapter Class Reference

```
#include <InputEngineAdapter.hpp>
```

Inheritance diagram for InputEngineAdapter:



Public Member Functions

- virtual Event< Input > & **getInputEvent** ()=0
- virtual Input **getInput** ()=0
- virtual void **getMousePosition** (int &x, int &y) const =0

16.59.1 Detailed Description

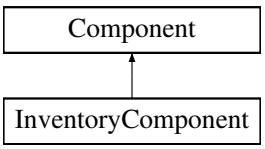
Input Adapter for user input.

The documentation for this class was generated from the following file:

- Engine/Input/Adapter/InputEngineAdapter.hpp

16.60 InventoryComponent Class Reference

Inheritance diagram for InventoryComponent:



Public Member Functions

- **InventoryComponent** (EntityId id)
- void **render** () override
- void **fixedUpdate** (const float &deltaTime) override
- std::string **name** () const override
- Component * **build** (EntityId entityId, const Components::component *component) override
- void **initialize** (EntityObject &entityParent) override
- void **update** (const Input &inputSystem) override
- void **addToInventory** (InventoryItem *item)
- void **removeFromInventory** (const std::string &name, int count)
- int **getInventorySize** () const
- bool **isMenuOpen** () const
- Event< InventoryItem & > & **getEventManager** ()

Additional Inherited Members

The documentation for this class was generated from the following files:

- Game/Components/Inventory/InventoryComponent.hpp
- Game/Components/Inventory/InventoryComponent.cpp

16.61 InventoryItem Class Reference

Public Types

- enum **itemType** { **weapon**, **resource**, **object** }

Public Member Functions

- **InventoryItem** (int quantity, std::string name, itemType type)
- void **onClick** ()
- std::string **getName** ()
- Vector2 & **getPosition** ()
- Vector2 & **getIndex** ()
- int **getItemQuantity** () const
- int **addItemQuantity** (int quantity)
- void **setIndex** (Vector2 &value)
- void **setPosition** (Vector2 &value)

The documentation for this class was generated from the following files:

- Game/Components/Inventory/InventoryItem.hpp
- Game/Components/Inventory/InventoryItem.cpp

16.62 KeyMap Class Reference

`#include <KeyMap.hpp>`

Static Public Attributes

- static std::map< SDL_Keycode, InputAction > mouseMap
- static std::map< SDL_Keycode, InputAction > controllerMap
- static std::map< SDL_Keycode, InputAction > keyboardMap

16.62.1 Detailed Description

Contains mappings for mouse, controller and keyboard. A map contains a SDL_KeyCode and an InputAction struct.

16.62.2 Member Data Documentation

16.62.2.1 controllerMap

```
std::map<SDL_Keycode, InputAction> KeyMap::controllerMap  [inline], [static]
```

Initial value:

```
= {
            {SDL_CONTROLLER_BUTTON_A,             InputAction{.code = "CONTROLLER_BUTTON_A", .action =
      "JUMP"}},
            {SDL_CONTROLLER_BUTTON_B,             InputAction{.code = "CONTROLLER_BUTTON_B", .action = ""}},
            {SDL_CONTROLLER_BUTTON_X,             InputAction{.code = "CONTROLLER_BUTTON_X", .action = ""}},
            {SDL_CONTROLLER_BUTTON_Y,             InputAction{.code = "CONTROLLER_BUTTON_Y", .action = ""}},
            {SDL_CONTROLLER_BUTTON_LEFTSHOULDER,  InputAction{.code = "CONTROLLER_BUTTON_LB", .action =
      ""}},
            {SDL_CONTROLLER_BUTTON_RIGHTSHOULDER, InputAction{.code = "CONTROLLER_BUTTON_RB", .action =
      ""}},
            {SDL_CONTROLLER_BUTTON_BACK,          InputAction{.code = "CONTROLLER_BUTTON_SELECT", .action =
      "QUIT"}},
            {SDL_CONTROLLER_BUTTON_START,         InputAction{.code = "CONTROLLER_BUTTON_START", .action =
      ""}},
            {SDL_CONTROLLER_BUTTON_LEFTSTICK,     InputAction{.code = "CONTROLLER_BUTTON_L3", .action =
      ""}},
            {SDL_CONTROLLER_BUTTON_RIGHTSTICK,    InputAction{.code = "CONTROLLER_BUTTON_R3", .action =
      ""}},
            {SDL_CONTROLLER_BUTTON_DPAD_UP,       InputAction{.code = "CONTOLLER_DPAD_UP", .action = "UP"}},
            {SDL_CONTROLLER_BUTTON_DPAD_DOWN,     InputAction{.code = "CONTOLLER_DPAD_DOWN", .action =
      "DOWN"}},
            {SDL_CONTROLLER_BUTTON_DPAD_LEFT,     InputAction{.code = "CONTOLLER_DPAD_LEFT", .action =
      "LEFT"}},
            {SDL_CONTROLLER_BUTTON_DPAD_RIGHT,    InputAction{.code = "CONTOLLER_DPAD_RIGHT", .action =
      "RIGHT"}},
    }
```

Keymap for device 2: controller

16.62.2.2 keyboardMap

```
std::map<SDL_Keycode, InputAction> KeyMap::keyboardMap  [inline], [static]
```

Initial value:

```
= {
            {SDLK_1,            InputAction{.code = "1", .action = "1"}},
            {SDLK_2,            InputAction{.code = "2", .action = "2"}},
            {SDLK_3,            InputAction{.code = "3", .action = "3"}},
            {SDLK_UP,           InputAction{.code = "UP", .action = "UP"}},
            {SDLK_DOWN,         InputAction{.code = "DOWN", .action = "DOWN"}},
            {SDLK_LEFT,         InputAction{.code = "LEFT", .action = "LEFT"}},
            {SDLK_RIGHT,        InputAction{.code = "RIGHT", .action = "RIGHT"}},
            {SDLK_w,            InputAction{.code = "W", .action = "UP"}},
            {SDLK_s,            InputAction{.code = "S", .action = "DOWN"}},
            {SDLK_a,            InputAction{.code = "A", .action = "LEFT"}},
            {SDLK_d,            InputAction{.code = "D", .action = "RIGHT"}},
            {SDLK_e,            InputAction{.code = "E", .action = "INTERACT"}},
            {SDLK_SPACE,        InputAction{.code = "SPACE", .action = "JUMP"}},
            {SDLK_i,            InputAction{.code = "I", .action = "INVENTORY"}},
            {SDLK_q,            InputAction{.code = "Q", .action = "QUIT"}},
            {SDLK_BACKSLASH,    InputAction{.code = "\\", .action = "BACKSLASH"}},
            {SDLK_RIGHTBRACKET, InputAction{.code = "]", .action = "RIGHTBRACKET"}},
            {SDLK_ESCAPE,       InputAction{.code = "ESC", .action = "ESCAPE"}},
    }
```

Keymap for device 0: keyboard

16.62.2.3 mouseMap

```
std::map<SDL_Keycode, InputAction> KeyMap::mouseMap  [inline], [static]
```

Initial value:

```
= {
            {SDL_BUTTON_LEFT,  InputAction{.code = "MOUSE_BUTTON_LEFT", .action = "CLICK_LEFT"}},
            {SDL_BUTTON_RIGHT, InputAction{.code = "MOUSE_BUTTON_RIGHT", .action = "CLICK_RIGHT"}},
    }
```

Keymap for device 1: mouse

The documentation for this class was generated from the following file:

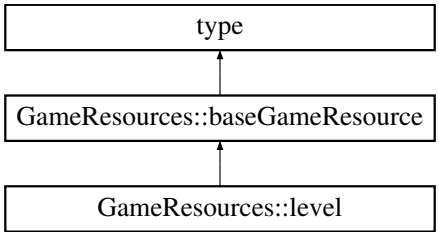
- Engine/Input/KeyMap.hpp

16.63 GameResources::level Class Reference

Class corresponding to the level schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::level:



spriteName

Accessor and modifier functions for the spriteName required element.

- typedef ::xml_schema::string spriteName_type
  Element type.
- typedef ::xsd::cxx::tree::traits< spriteName_type, char > spriteName_traits
  Element traits type.
- const spriteName_type & spriteName () const
  Return a read-only (constant) reference to the element.
- spriteName_type & spriteName ()
  Return a read-write reference to the element.
- void spriteName (const spriteName_type &x)
  Set the element value.
- void spriteName (::std::unique_ptr< spriteName_type > p)
  Set the element value without copying.

spriteSheetPath

Accessor and modifier functions for the spriteSheetPath required element.

- typedef ::xml_schema::string spriteSheetPath_type
  Element type.
- typedef ::xsd::cxx::tree::traits< spriteSheetPath_type, char > spriteSheetPath_traits
  Element traits type.
- const spriteSheetPath_type & spriteSheetPath () const
  Return a read-only (constant) reference to the element.
- spriteSheetPath_type & spriteSheetPath ()
  Return a read-write reference to the element.
- void spriteSheetPath (const spriteSheetPath_type &x)
  Set the element value.
- void spriteSheetPath (::std::unique_ptr< spriteSheetPath_type > p)
  Set the element value without copying.

tmxPath

Accessor and modifier functions for the tmxPath required element.

- typedef ::xml_schema::string tmxPath_type

  Element type.
- typedef ::xsd::cxx::tree::traits< tmxPath_type, char > tmxPath_traits

  Element traits type.
- const tmxPath_type & tmxPath () const

  Return a read-only (constant) reference to the element.
- tmxPath_type & tmxPath ()

  Return a read-write reference to the element.
- void tmxPath (const tmxPath_type &x)

  Set the element value.
- void tmxPath (::std::unique_ptr< tmxPath_type > p)

  Set the element value without copying.

Constructors

- level (const name_type &, const path_type &, const spriteName_type &, const spriteSheetPath_type &, const tmxPath_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- level (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- level (const level &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual level * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- level & operator= (const level &x)

  Copy assignment operator.
- virtual ~level ()

  Destructor.

Additional Inherited Members

16.63.1 Detailed Description

Class corresponding to the level schema type.

16.63.2 Constructor & Destructor Documentation

16.63.2.1 level() [1/2]

```
GameResources::level::level (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.63.2.2 level() [2/2]

```
GameResources::level::level (
            const level & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.63.3 Member Function Documentation

16.63.3.1 _clone()

```
level * GameResources::level::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

Reimplemented from GameResources::baseGameResource.

16.63.3.2 operator=()

```
level & GameResources::level::operator= (
            const level & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.63.3.3 spriteName() [1/4]

```
level::spriteName_type & GameResources::level::spriteName ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.63.3.4 spriteName() [2/4]

```
const level::spriteName_type & GameResources::level::spriteName ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.63.3.5 spriteName() [3/4]

```
void GameResources::level::spriteName (
            ::std::unique_ptr< spriteName_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.63.3.6 spriteName() [4/4]

```
void GameResources::level::spriteName (
            const spriteName_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.63.3.7 spriteSheetPath() [1/4]

```
level::spriteSheetPath_type & GameResources::level::spriteSheetPath ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.63.3.8 spriteSheetPath() [2/4]

```
const level::spriteSheetPath_type & GameResources::level::spriteSheetPath ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.63.3.9 spriteSheetPath() [3/4]

```
void GameResources::level::spriteSheetPath (
            ::std::unique_ptr< spriteSheetPath_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.63.3.10 spriteSheetPath() [4/4]

```
void GameResources::level::spriteSheetPath (
            const spriteSheetPath_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.63.3.11 tmxPath() [1/4]

```
level::tmxPath_type & GameResources::level::tmxPath ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.63.3.12 tmxPath() [2/4]

```
const level::tmxPath_type & GameResources::level::tmxPath ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.63.3.13 tmxPath() [3/4]

```
void GameResources::level::tmxPath (
            ::std::unique_ptr< tmxPath_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.63.3.14 tmxPath() [4/4]

```
void GameResources::level::tmxPath (
            const tmxPath_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

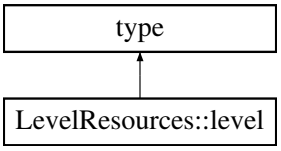
- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.64 LevelResources::level Class Reference

Class corresponding to the level schema type.

```
#include <level-resources.hxx>
```

Inheritance diagram for LevelResources::level:



name

Accessor and modifier functions for the name required element.

- typedef ::xml_schema::string name_type

  Element type.
- typedef ::xsd::cxx::tree::traits< name_type, char > name_traits

  Element traits type.
- const name_type & name () const

Return a read-only (constant) reference to the element.

- name_type & name ()

  Return a read-write reference to the element.

- void name (const name_type &x)

  Set the element value.

- void name (::std::unique_ptr< name_type > p)

  Set the element value without copying.

object

Accessor and modifier functions for the object sequence element.

- typedef ::Objects::object object_type

  Element type.

- typedef ::xsd::cxx::tree::sequence< object_type > object_sequence

  Element sequence container type.

- typedef object_sequence::iterator object_iterator

  Element iterator type.

- typedef object_sequence::const_iterator object_const_iterator

  Element constant iterator type.

- typedef ::xsd::cxx::tree::traits< object_type, char > object_traits

  Element traits type.

- const object_sequence & object () const

  Return a read-only (constant) reference to the element sequence.

- object_sequence & object ()

  Return a read-write reference to the element sequence.

- void object (const object_sequence &s)

  Copy elements from a given sequence.

Constructors

- level (const name_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.

- level (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.

- level (const level &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.

- virtual level * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.

- level & operator= (const level &x)

  Copy assignment operator.

- virtual ~level ()

  Destructor.

16.64.1 Detailed Description

Class corresponding to the level schema type.

16.64.2 Constructor & Destructor Documentation

16.64.2.1 level() [1/2]

```
LevelResources::level::level (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.64.2.2 level() [2/2]

```
LevelResources::level::level (
            const level & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.64.3 Member Function Documentation

16.64.3.1 _clone()

```
level * LevelResources::level::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.64.3.2 name() [1/4]

```
level::name_type & LevelResources::level::name ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.64.3.3 name() [2/4]

```
const level::name_type & LevelResources::level::name ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.64.3.4 name() [3/4]

```
void LevelResources::level::name (
            ::std::unique_ptr< name_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.64.3.5 name() [4/4]

```
void LevelResources::level::name (
            const name_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.64.3.6 object() [1/3]

```
level::object_sequence & LevelResources::level::object ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.64.3.7 object() [2/3]

```
const level::object_sequence & LevelResources::level::object ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.64.3.8 object() [3/3]

```
void LevelResources::level::object (
            const object_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

16.64.3.9 operator=()

```
level & LevelResources::level::operator= (
            const level & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

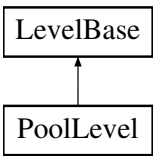
For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/level-resources.hxx
- Resources/XML/Generated/level-resources.cxx

16.65 LevelBase Class Reference

Inheritance diagram for LevelBase:



Public Member Functions

- virtual void **initialize** (const std::string &name, const LevelData &data)
- virtual void **render** ()
- virtual void **update** (const Input &inputSystem)
- virtual void **fixedUpdate** (float deltaTime)
- virtual void **clearEntities** ()

Public Attributes

- CharacterComponent * **_characterComponent** = nullptr

The documentation for this class was generated from the following files:

- Game/Scenes/LevelBase.hpp
- Game/Scenes/LevelBase.cpp

16.66 LevelData Struct Reference

Public Member Functions

- **LevelData** (const std::string _tmxPath, const std::string spriteSheetPath, const std::string &spriteId, const std::string &levelResourcePath)

Public Attributes

- const std::string **tmxPath**
- const std::string **spritesheetPath**
- const std::string **spriteId**
- const std::string **levelResourcePath**

The documentation for this struct was generated from the following file:

- API/Rendering/RenderingAPI.hpp

16.67 LevelParserAPI Class Reference

Static Public Member Functions

- static void **loadEntities** (std::multimap< std::string, Components::component * > &outEntities, xsd::cxx↩ ::tree::sequence< Objects::object > &objects)
- static TMXLevel * **loadLevel** (std::multimap< std::string, Components::component * > &outEntities, const LevelData &levelData)

The documentation for this class was generated from the following files:

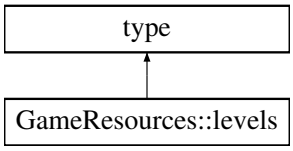
- API/XMLParser/LevelParserAPI.hpp
- API/XMLParser/LevelParserAPI.cpp

16.68 GameResources::levels Class Reference

Class corresponding to the levels schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::levels:



level

Accessor and modifier functions for the level sequence element.

- typedef ::GameResources::level level_type
  Element type.
- typedef ::xsd::cxx::tree::sequence< level_type > level_sequence
  Element sequence container type.
- typedef level_sequence::iterator level_iterator
  Element iterator type.
- typedef level_sequence::const_iterator level_const_iterator
  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< level_type, char > level_traits
  Element traits type.
- const level_sequence & level () const
  Return a read-only (constant) reference to the element sequence.
- level_sequence & level ()
  Return a read-write reference to the element sequence.
- void level (const level_sequence &s)
  Copy elements from a given sequence.

Constructors

- levels ()
  Create an instance from the ultimate base and initializers for required elements and attributes.
- levels (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- levels (const levels &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual levels * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- levels & operator= (const levels &x)
  Copy assignment operator.
- virtual ~levels ()
  Destructor.

16.68.1 Detailed Description

Class corresponding to the levels schema type.

16.68.2 Constructor & Destructor Documentation

16.68.2.1 levels() [1/2]

```
GameResources::levels::levels (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.68.2.2 levels() [2/2]

```
GameResources::levels::levels (
            const levels & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.68.3 Member Function Documentation

16.68.3.1 _clone()

```
levels * GameResources::levels::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.68.3.2 level() [1/3]

```
levels::level_sequence & GameResources::levels::level ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.68.3.3 level() [2/3]

```
const levels::level_sequence & GameResources::levels::level ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.68.3.4 level() [3/3]

```
void GameResources::levels::level (
            const level_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

16.68.3.5 operator=()

```
levels & GameResources::levels::operator= (
            const levels & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.69 LoadedObjectData Struct Reference

Public Attributes

- std::string **objectName**
- Vector2 **position**

The documentation for this struct was generated from the following file:

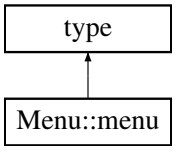
- Engine/Rendering/TMXLevel.hpp

16.70 Menu::menu Class Reference

Class corresponding to the menu schema type.

`#include <menu.hxx>`

Inheritance diagram for Menu::menu:



name

Accessor and modifier functions for the name required element.

- typedef ::xml_schema::string name_type

  Element type.
- typedef ::xsd::cxx::tree::traits< name_type, char > name_traits

  Element traits type.
- const name_type & name () const

  Return a read-only (constant) reference to the element.
- name_type & name ()

  Return a read-write reference to the element.
- void name (const name_type &x)

  Set the element value.
- void name (::std::unique_ptr< name_type > p)

  Set the element value without copying.

resources

Accessor and modifier functions for the resources optional element.

- typedef ::Common::resources resources_type

  Element type.
- typedef ::xsd::cxx::tree::optional< resources_type > resources_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< resources_type, char > resources_traits

  Element traits type.
- const resources_optional & resources () const

  Return a read-only (constant) reference to the element container.
- resources_optional & resources ()

  Return a read-write reference to the element container.
- void resources (const resources_type &x)

  Set the element value.
- void resources (const resources_optional &x)

  Set the element value.
- void resources (::std::unique_ptr< resources_type > p)

  Set the element value without copying.

color

Accessor and modifier functions for the color optional element.

- typedef ::Common::color color_type

  Element type.
- typedef ::xsd::cxx::tree::optional< color_type > color_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< color_type, char > color_traits

Element traits type.

- const color_optional & color () const

  Return a read-only (constant) reference to the element container.
- color_optional & color ()

  Return a read-write reference to the element container.
- void color (const color_type &x)

  Set the element value.
- void color (const color_optional &x)

  Set the element value.
- void color (::std::unique_ptr< color_type > p)

  Set the element value without copying.

backgroundMusic

Accessor and modifier functions for the backgroundMusic optional element.

- typedef ::xml_schema::string backgroundMusic_type

  Element type.
- typedef ::xsd::cxx::tree::optional< backgroundMusic_type > backgroundMusic_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< backgroundMusic_type, char > backgroundMusic_traits

  Element traits type.
- const backgroundMusic_optional & backgroundMusic () const

  Return a read-only (constant) reference to the element container.
- backgroundMusic_optional & backgroundMusic ()

  Return a read-write reference to the element container.
- void backgroundMusic (const backgroundMusic_type &x)

  Set the element value.
- void backgroundMusic (const backgroundMusic_optional &x)

  Set the element value.
- void backgroundMusic (::std::unique_ptr< backgroundMusic_type > p)

  Set the element value without copying.

buttons

Accessor and modifier functions for the buttons optional element.

- typedef ::Menu::buttons buttons_type

  Element type.
- typedef ::xsd::cxx::tree::optional< buttons_type > buttons_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< buttons_type, char > buttons_traits

  Element traits type.
- const buttons_optional & buttons () const

  Return a read-only (constant) reference to the element container.
- buttons_optional & buttons ()

  Return a read-write reference to the element container.
- void buttons (const buttons_type &x)

  Set the element value.
- void buttons (const buttons_optional &x)

  Set the element value.
- void buttons (::std::unique_ptr< buttons_type > p)

  Set the element value without copying.

texts

Accessor and modifier functions for the texts required element.

- typedef ::Menu::texts texts_type
  Element type.
- typedef ::xsd::cxx::tree::traits< texts_type, char > texts_traits
  Element traits type.
- const texts_type & texts () const
  Return a read-only (constant) reference to the element.
- texts_type & texts ()
  Return a read-write reference to the element.
- void texts (const texts_type &x)
  Set the element value.
- void texts (::std::unique_ptr< texts_type > p)
  Set the element value without copying.

images

Accessor and modifier functions for the images optional element.

- typedef ::Menu::images images_type
  Element type.
- typedef ::xsd::cxx::tree::optional< images_type > images_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< images_type, char > images_traits
  Element traits type.
- const images_optional & images () const
  Return a read-only (constant) reference to the element container.
- images_optional & images ()
  Return a read-write reference to the element container.
- void images (const images_type &x)
  Set the element value.
- void images (const images_optional &x)
  Set the element value.
- void images (::std::unique_ptr< images_type > p)
  Set the element value without copying.

boxes

Accessor and modifier functions for the boxes optional element.

- typedef ::Menu::boxes boxes_type
  Element type.
- typedef ::xsd::cxx::tree::optional< boxes_type > boxes_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< boxes_type, char > boxes_traits

Element traits type.

- const boxes_optional & boxes () const

  Return a read-only (constant) reference to the element container.
- boxes_optional & boxes ()

  Return a read-write reference to the element container.
- void boxes (const boxes_type &x)

  Set the element value.
- void boxes (const boxes_optional &x)

  Set the element value.
- void boxes (::std::unique_ptr< boxes_type > p)

  Set the element value without copying.

preloadResources

Accessor and modifier functions for the preloadResources required element.

- typedef ::Common::preloadResources preloadResources_type

  Element type.
- typedef ::xsd::cxx::tree::traits< preloadResources_type, char > preloadResources_traits

  Element traits type.
- const preloadResources_type & preloadResources () const

  Return a read-only (constant) reference to the element.
- preloadResources_type & preloadResources ()

  Return a read-write reference to the element.
- void preloadResources (const preloadResources_type &x)

  Set the element value.
- void preloadResources (::std::unique_ptr< preloadResources_type > p)

  Set the element value without copying.

Constructors

- menu (const name_type &, const texts_type &, const preloadResources_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- menu (const name_type &, ::std::unique_ptr< texts_type >, ::std::unique_ptr< preloadResources_type >)

  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- menu (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- menu (const menu &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual menu * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- menu & operator= (const menu &x)

  Copy assignment operator.
- virtual ~menu ()

  Destructor.

16.70.1 Detailed Description

Class corresponding to the menu schema type.

16.70.2 Constructor & Destructor Documentation

16.70.2.1 menu() [1/3]

```
Menu::menu::menu (
            const name_type & name,
            ::std::unique_ptr< texts_type > texts,
            ::std::unique_ptr< preloadResources_type > preloadResources )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.70.2.2 menu() [2/3]

```
Menu::menu::menu (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.70.2.3 menu() [3/3]

```
Menu::menu::menu (
            const menu & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.70.3 Member Function Documentation

16.70.3.1 _clone()

```
menu * Menu::menu::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.70.3.2 backgroundMusic() [1/5]

```
menu::backgroundMusic_optional & Menu::menu::backgroundMusic ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.70.3.3 backgroundMusic() [2/5]

```
const menu::backgroundMusic_optional & Menu::menu::backgroundMusic ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.70.3.4 backgroundMusic() [3/5]

```
void Menu::menu::backgroundMusic (
            ::std::unique_ptr< backgroundMusic_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.70.3.5 backgroundMusic() [4/5]

```
void Menu::menu::backgroundMusic (
            const backgroundMusic_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.70.3.6 backgroundMusic() [5/5]

```
void Menu::menu::backgroundMusic (
            const backgroundMusic_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.70.3.7 boxes() [1/5]

```
menu::boxes_optional & Menu::menu::boxes ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.70.3.8 boxes() [2/5]

```
const menu::boxes_optional & Menu::menu::boxes ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.70.3.9 boxes() [3/5]

```
void Menu::menu::boxes (
            ::std::unique_ptr< boxes_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.70.3.10 boxes() [4/5]

```
void Menu::menu::boxes (
            const boxes_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.70.3.11 boxes() [5/5]

```
void Menu::menu::boxes (
            const boxes_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.70.3.12 buttons() [1/5]

```
menu::buttons_optional & Menu::menu::buttons ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.70.3.13 buttons() [2/5]

```
const menu::buttons_optional & Menu::menu::buttons ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.70.3.14 buttons() [3/5]

```
void Menu::menu::buttons (
            ::std::unique_ptr< buttons_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.70.3.15 buttons() [4/5]

```
void Menu::menu::buttons (
            const buttons_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.70.3.16 buttons() [5/5]

```
void Menu::menu::buttons (
            const buttons_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.70.3.17 color() [1/5]

```
menu::color_optional & Menu::menu::color ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.70.3.18 color() [2/5]

```
const menu::color_optional & Menu::menu::color ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.70.3.19 color() [3/5]

```
void Menu::menu::color (
            ::std::unique_ptr< color_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.70.3.20 color() [4/5]

```
void Menu::menu::color (
            const color_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.70.3.21 color() [5/5]

```
void Menu::menu::color (
            const color_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.70.3.22 images() [1/5]

```
menu::images_optional & Menu::menu::images ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.70.3.23 images() [2/5]

```
const menu::images_optional & Menu::menu::images ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.70.3.24 images() [3/5]

```
void Menu::menu::images (
            ::std::unique_ptr< images_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.70.3.25 images() [4/5]

```
void Menu::menu::images (
            const images_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.70.3.26 images() [5/5]

```
void Menu::menu::images (
            const images_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.70.3.27 name() [1/4]

```
menu::name_type & Menu::menu::name ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.70.3.28 name() [2/4]

```
const menu::name_type & Menu::menu::name ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.70.3.29 name() [3/4]

```
void Menu::menu::name (
            ::std::unique_ptr< name_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.70.3.30 name() [4/4]

```
void Menu::menu::name (
            const name_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.70.3.31 operator=()

```
menu & Menu::menu::operator= (
            const menu & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.70.3.32 preloadResources() [1/4]

```
menu::preloadResources_type & Menu::menu::preloadResources ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.70.3.33 preloadResources() [2/4]

```
const menu::preloadResources_type & Menu::menu::preloadResources ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.70.3.34 preloadResources() [3/4]

```
void Menu::menu::preloadResources (
            ::std::unique_ptr< preloadResources_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.70.3.35 preloadResources() [4/4]

```
void Menu::menu::preloadResources (
            const preloadResources_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.70.3.36 resources() [1/5]

```
menu::resources_optional & Menu::menu::resources ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.70.3.37 resources() [2/5]

```
const menu::resources_optional & Menu::menu::resources ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.70.3.38 resources() [3/5]

```
void Menu::menu::resources (
            ::std::unique_ptr< resources_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.70.3.39 resources() [4/5]

```
void Menu::menu::resources (
            const resources_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.70.3.40 resources() [5/5]

```
void Menu::menu::resources (
            const resources_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.70.3.41 texts() [1/4]

```
menu::texts_type & Menu::menu::texts ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.70.3.42 texts() [2/4]

```
const menu::texts_type & Menu::menu::texts ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.70.3.43 texts() [3/4]

```
void Menu::menu::texts (
            ::std::unique_ptr< texts_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.70.3.44 texts() [4/4]

```
void Menu::menu::texts (
            const texts_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.71 MenuParser Class Reference

Public Member Functions

- **MenuParser** (MenuParser &other)=delete
- void **operator=** (const MenuParser &)=delete
- **MenuParser** (const RenderingAPI &renderer)
- void **initialize** (const std::string &path)
- void **render** ()
- void **onClick** (const Input &input)
- Event< std::string > & **getCustomEventHandler** ()

Static Public Member Functions

- static MenuParser * **getInstance** ()

Public Attributes

- std::stack< std::string > **PreviousScenes**

The documentation for this class was generated from the following files:

- Engine/XMLParser/MenuParser.hpp
- Engine/XMLParser/MenuParser.cpp

16.72 MenuParserAPI Class Reference

Public Member Functions

- **MenuParserAPI** (RenderingAPI &renderer, Event< Input > &event)
- void **loadScene** (std::string path)
- void **render** ()
- void **onClick** (Input input)
- Event< std::string > & **getCustomEventHandler** ()

The documentation for this class was generated from the following files:

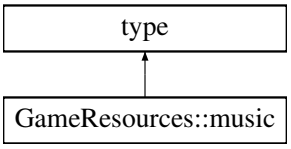
- API/XMLParser/MenuParserAPI.hpp
- API/XMLParser/MenuParserAPI.cpp

16.73 GameResources::music Class Reference

Class corresponding to the music schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::music:



music

Accessor and modifier functions for the music sequence element.

- typedef ::GameResources::music1 music1_type
  Element type.
- typedef ::xsd::cxx::tree::sequence< music1_type > music1_sequence
  Element sequence container type.
- typedef music1_sequence::iterator music1_iterator
  Element iterator type.
- typedef music1_sequence::const_iterator music1_const_iterator
  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< music1_type, char > music1_traits
  Element traits type.
- const music1_sequence & music1 () const
  Return a read-only (constant) reference to the element sequence.
- music1_sequence & music1 ()
  Return a read-write reference to the element sequence.
- void music1 (const music1_sequence &s)
  Copy elements from a given sequence.

Constructors

- music ()
  Create an instance from the ultimate base and initializers for required elements and attributes.
- music (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- music (const music &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual music * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- music & operator= (const music &x)
  Copy assignment operator.
- virtual ~music ()
  Destructor.

16.73.1 Detailed Description

Class corresponding to the music schema type.

16.73.2 Constructor & Destructor Documentation

16.73.2.1 music() [1/2]

```
GameResources::music::music (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.73.2.2 music() [2/2]

```
GameResources::music::music (
            const music & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.73.3 Member Function Documentation

16.73.3.1 _clone()

```
music * GameResources::music::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.73.3.2 music1() [1/3]

```
music::music1_sequence & GameResources::music::music1 ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.73.3.3 music1() [2/3]

```
const music::music1_sequence & GameResources::music::music1 ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.73.3.4 music1() [3/3]

```
void GameResources::music::music1 (
            const music1_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

16.73.3.5 operator=()

```
music & GameResources::music::operator= (
            const music & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

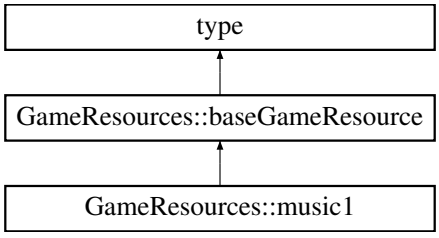
- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.74 GameResources::music1 Class Reference

Class corresponding to the music1 schema type.

```
#include <resources.hxx>
```

Inheritance diagram for GameResources::music1:



Constructors

- music1 (const name_type &, const path_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- music1 (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- music1 (const music1 &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual music1 * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- virtual ~music1 ()

  Destructor.

Additional Inherited Members

16.74.1 Detailed Description

Class corresponding to the music1 schema type.

16.74.2 Constructor & Destructor Documentation

16.74.2.1 music1() [1/2]

```
GameResources::music1::music1 (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.74.2.2 music1() [2/2]

```
GameResources::music1::music1 (
            const music1 & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.74.3 Member Function Documentation

16.74.3.1 _clone()

```
music1 * GameResources::music1::_clone (
             ::xml_schema::flags f = 0,
             ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

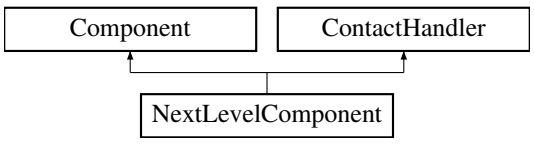
Reimplemented from GameResources::baseGameResource.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.75 NextLevelComponent Class Reference

Inheritance diagram for NextLevelComponent:



Public Member Functions

- **NextLevelComponent** (EntityId id)
- void **startContact** (b2Contact ∗contact) override
- void **endContact** (b2Contact ∗contact) override
- void **render** () override
- void **update** (const Input &inputSystem) override
- void **fixedUpdate** (const float &deltaTime) override
- Component ∗ **build** (EntityId entityId, const Components::component ∗component) override
- void **initialize** (EntityObject &entityParent) override
- std::string **name** () const override

Public Attributes

- std::string **NextLevel**
- bool **hasContactWithPlayer** = false

Additional Inherited Members

The documentation for this class was generated from the following files:

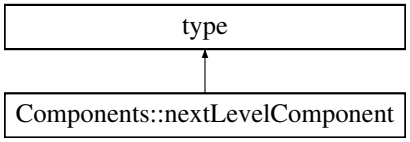
- Game/Components/NextLevelComponent.hpp
- Game/Components/NextLevelComponent.cpp

16.76 Components::nextLevelComponent Class Reference

Class corresponding to the nextLevelComponent schema type.

`#include <components.hxx>`

Inheritance diagram for Components::nextLevelComponent:



levelName

Accessor and modifier functions for the levelName required element.

- typedef ::xml_schema::string levelName_type
  Element type.
- typedef ::xsd::cxx::tree::traits< levelName_type, char > levelName_traits
  Element traits type.
- const levelName_type & levelName () const
  Return a read-only (constant) reference to the element.
- levelName_type & levelName ()
  Return a read-write reference to the element.
- void levelName (const levelName_type &x)
  Set the element value.
- void levelName (::std::unique_ptr< levelName_type > p)
  Set the element value without copying.

Constructors

- nextLevelComponent (const levelName_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- nextLevelComponent (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- nextLevelComponent (const nextLevelComponent &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual nextLevelComponent * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- nextLevelComponent & operator= (const nextLevelComponent &x)

  Copy assignment operator.
- virtual ~nextLevelComponent ()

  Destructor.

16.76.1 Detailed Description

Class corresponding to the nextLevelComponent schema type.

16.76.2 Constructor & Destructor Documentation

16.76.2.1 nextLevelComponent() [1/2]

```
Components::nextLevelComponent::nextLevelComponent (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.76.2.2 nextLevelComponent() [2/2]

```
Components::nextLevelComponent::nextLevelComponent (
            const nextLevelComponent & x,
```

```
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.76.3 Member Function Documentation

16.76.3.1 _clone()

```
nextLevelComponent * Components::nextLevelComponent::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.76.3.2 levelName() [1/4]

```
nextLevelComponent::levelName_type & Components::nextLevelComponent::levelName ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.76.3.3 levelName() [2/4]

```
const nextLevelComponent::levelName_type & Components::nextLevelComponent::levelName ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.76.3.4 levelName() [3/4]

```
void Components::nextLevelComponent::levelName (
            ::std::unique_ptr< levelName_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.76.3.5 levelName() [4/4]

```
void Components::nextLevelComponent::levelName (
            const levelName_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.76.3.6 operator=()

```
nextLevelComponent & Components::nextLevelComponent::operator= (
            const nextLevelComponent & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

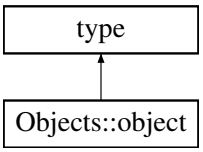
- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.77 Objects::object Class Reference

Class corresponding to the object schema type.

`#include <objects.hxx>`

Inheritance diagram for Objects::object:



name

Accessor and modifier functions for the name required element.

- typedef ::xml_schema::string name_type

  Element type.
- typedef ::xsd::cxx::tree::traits< name_type, char > name_traits

  Element traits type.
- const name_type & name () const

  Return a read-only (constant) reference to the element.
- name_type & name ()

  Return a read-write reference to the element.
- void name (const name_type &x)

  Set the element value.
- void name (::std::unique_ptr< name_type > p)

  Set the element value without copying.

components

Accessor and modifier functions for the components required element.

- typedef ::Objects::components components_type

  Element type.
- typedef ::xsd::cxx::tree::traits< components_type, char > components_traits

  Element traits type.
- const components_type & components () const

  Return a read-only (constant) reference to the element.
- components_type & components ()

  Return a read-write reference to the element.
- void components (const components_type &x)

  Set the element value.
- void components (::std::unique_ptr< components_type > p)

  Set the element value without copying.

Constructors

- object (const name_type &, const components_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- object (const name_type &, ::std::unique_ptr< components_type >)

  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- object (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- object (const object &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual object * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- object & operator= (const object &x)

  Copy assignment operator.
- virtual ~object ()

  Destructor.

16.77.1 Detailed Description

Class corresponding to the object schema type.

16.77.2 Constructor & Destructor Documentation

16.77.2.1 object() [1/3]

```
Objects::object::object (
            const name_type & name,
            ::std::unique_ptr< components_type > components )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.77.2.2 object() [2/3]

```
Objects::object::object (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.77.2.3 object() [3/3]

```
Objects::object::object (
            const object & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.77.3 Member Function Documentation

16.77.3.1 _clone()

```
object * Objects::object::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.77.3.2 components() [1/4]

```
object::components_type & Objects::object::components ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.77.3.3 components() [2/4]

```
const object::components_type & Objects::object::components ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.77.3.4 components() [3/4]

```
void Objects::object::components (
            ::std::unique_ptr< components_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.77.3.5 components() [4/4]

```
void Objects::object::components (
            const components_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.77.3.6 name() [1/4]

```
object::name_type & Objects::object::name ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.77.3.7 name() [2/4]

```
const object::name_type & Objects::object::name ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.77.3.8 name() [3/4]

```
void Objects::object::name (
            ::std::unique_ptr< name_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.77.3.9 name() [4/4]

```
void Objects::object::name (
            const name_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.77.3.10 operator=()

```
object & Objects::object::operator= (
            const object & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

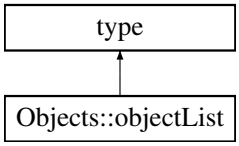
- Resources/XML/Generated/objects.hxx
- Resources/XML/Generated/objects.cxx

16.78 Objects::objectList Class Reference

Class corresponding to the objectList schema type.

```
#include <objects.hxx>
```

Inheritance diagram for Objects::objectList:



object

Accessor and modifier functions for the object sequence element.

- typedef ::Objects::object object_type

  Element type.
- typedef ::xsd::cxx::tree::sequence< object_type > object_sequence

  Element sequence container type.
- typedef object_sequence::iterator object_iterator

  Element iterator type.
- typedef object_sequence::const_iterator object_const_iterator

  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< object_type, char > object_traits

  Element traits type.
- const object_sequence & object () const

  Return a read-only (constant) reference to the element sequence.
- object_sequence & object ()

  Return a read-write reference to the element sequence.
- void object (const object_sequence &s)

  Copy elements from a given sequence.

Constructors

- objectList ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- objectList (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- objectList (const objectList &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual objectList * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- objectList & operator= (const objectList &x)

  Copy assignment operator.
- virtual ~objectList ()

  Destructor.

16.78.1 Detailed Description

Class corresponding to the objectList schema type.

16.78.2 Constructor & Destructor Documentation

16.78.2.1 objectList() [1/2]

```
Objects::objectList::objectList (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.78.2.2 objectList() [2/2]

```
Objects::objectList::objectList (
            const objectList & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.78.3 Member Function Documentation

16.78.3.1 _clone()

```
objectList * Objects::objectList::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.78.3.2 object() [1/3]

```
objectList::object_sequence & Objects::objectList::object ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.78.3.3 object() [2/3]

```
const objectList::object_sequence & Objects::objectList::object ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.78.3.4 object() [3/3]

```
void Objects::objectList::object (
            const object_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

16.78.3.5 operator=()

```
objectList & Objects::objectList::operator= (
            const objectList & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

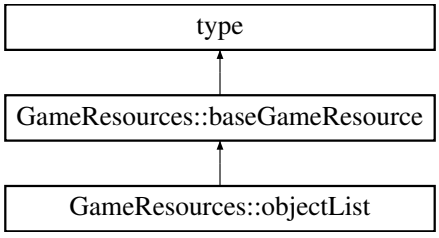
- Resources/XML/Generated/objects.hxx
- Resources/XML/Generated/objects.cxx

16.79 GameResources::objectList Class Reference

Class corresponding to the objectList schema type.

```
#include <resources.hxx>
```

Inheritance diagram for GameResources::objectList:



pool

Accessor and modifier functions for the pool required element.

- typedef ::GameResources::pool pool_type

  Element type.
- typedef ::xsd::cxx::tree::traits< pool_type, char > pool_traits

  Element traits type.
- const pool_type & pool () const

  Return a read-only (constant) reference to the element.
- pool_type & pool ()

  Return a read-write reference to the element.
- void pool (const pool_type &x)

  Set the element value.
- void pool (::std::unique_ptr< pool_type > p)

  Set the element value without copying.

Constructors

- objectList (const name_type &, const path_type &, const pool_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- objectList (const name_type &, const path_type &, ::std::unique_ptr< pool_type >)

  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- objectList (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- objectList (const objectList &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual objectList * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- objectList & operator= (const objectList &x)

  Copy assignment operator.
- virtual ~objectList ()

  Destructor.

Additional Inherited Members

16.79.1 Detailed Description

Class corresponding to the objectList schema type.

16.79.2 Constructor & Destructor Documentation

16.79.2.1 objectList() [1/3]

```
GameResources::objectList::objectList (
            const name_type & name,
            const path_type & path,
            ::std::unique_ptr< pool_type > pool )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.79.2.2 objectList() [2/3]

```
GameResources::objectList::objectList (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.79.2.3 objectList() [3/3]

```
GameResources::objectList::objectList (
            const objectList & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.79.3 Member Function Documentation

16.79.3.1 _clone()

```
objectList * GameResources::objectList::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

Reimplemented from GameResources::baseGameResource.

16.79.3.2 operator=()

```
objectList & GameResources::objectList::operator= (
            const objectList & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.79.3.3 pool() [1/4]

```
objectList::pool_type & GameResources::objectList::pool ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.79.3.4 pool() [2/4]

```
const objectList::pool_type & GameResources::objectList::pool ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.79.3.5 pool() [3/4]

```
void GameResources::objectList::pool (
            ::std::unique_ptr< pool_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.79.3.6 pool() [4/4]

```
void GameResources::objectList::pool (
            const pool_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

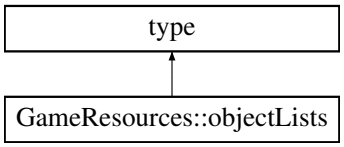
- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.80 GameResources::objectLists Class Reference

Class corresponding to the objectLists schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::objectLists:



objectList

Accessor and modifier functions for the objectList sequence element.

- typedef ::GameResources::objectList objectList_type
  Element type.
- typedef ::xsd::cxx::tree::sequence< objectList_type > objectList_sequence
  Element sequence container type.
- typedef objectList_sequence::iterator objectList_iterator
  Element iterator type.
- typedef objectList_sequence::const_iterator objectList_const_iterator
  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< objectList_type, char > objectList_traits
  Element traits type.
- const objectList_sequence & objectList () const
  Return a read-only (constant) reference to the element sequence.
- objectList_sequence & objectList ()
  Return a read-write reference to the element sequence.
- void objectList (const objectList_sequence &s)
  Copy elements from a given sequence.

Constructors

- objectLists ()
  Create an instance from the ultimate base and initializers for required elements and attributes.
- objectLists (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- objectLists (const objectLists &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual objectLists * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- objectLists & operator= (const objectLists &x)
  Copy assignment operator.
- virtual ~objectLists ()
  Destructor.

16.80.1 Detailed Description

Class corresponding to the objectLists schema type.

16.80.2 Constructor & Destructor Documentation

16.80.2.1 objectLists() [1/2]

```
GameResources::objectLists::objectLists (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.80.2.2 objectLists() [2/2]

```
GameResources::objectLists::objectLists (
            const objectLists & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.80.3 Member Function Documentation

16.80.3.1 _clone()

```
objectLists * GameResources::objectLists::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.80.3.2 objectList() [1/3]

```
objectLists::objectList_sequence & GameResources::objectLists::objectList ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.80.3.3 objectList() [2/3]

```
const objectLists::objectList_sequence & GameResources::objectLists::objectList ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.80.3.4 objectList() [3/3]

```
void GameResources::objectLists::objectList (
            const objectList_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

16.80.3.5 operator=()

```
objectLists & GameResources::objectLists::operator= (
            const objectLists & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.81 ObjectLoader Class Reference

Static Public Member Functions

- static void loadEntities (const std::multimap< std::string, Components::component * > &loadedEntities, std::vector< std::unique_ptr< EntityObject >> &entities)

16.81.1 Member Function Documentation

16.81.1.1 loadEntities()

```
void ObjectLoader::loadEntities (
            const std::multimap< std::string, Components::component * > & loadedEntities,
            std::vector< std::unique_ptr< EntityObject >> & entities )  [static]
```

ContactHandler

Contact Handlers & TransformComponent

The documentation for this class was generated from the following files:

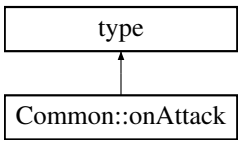
- Game/Object/ObjectLoader.hpp
- Game/Object/ObjectLoader.cpp

16.82 Common::onAttack Class Reference

Class corresponding to the onAttack schema type.

`#include <common.hxx>`

Inheritance diagram for Common::onAttack:



playSound

Accessor and modifier functions for the playSound optional element.

- typedef ::xml_schema::uri playSound_type
  Element type.
- typedef ::xsd::cxx::tree::optional< playSound_type > playSound_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< playSound_type, char > playSound_traits
  Element traits type.
- const playSound_optional & playSound () const
  Return a read-only (constant) reference to the element container.
- playSound_optional & playSound ()
  Return a read-write reference to the element container.
- void playSound (const playSound_type &x)
  Set the element value.
- void playSound (const playSound_optional &x)
  Set the element value.
- void playSound (::std::unique_ptr< playSound_type > p)
  Set the element value without copying.

loadAction

Accessor and modifier functions for the loadAction optional element.

- typedef ::xml_schema::uri loadAction_type
  Element type.
- typedef ::xsd::cxx::tree::optional< loadAction_type > loadAction_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< loadAction_type, char > loadAction_traits
  Element traits type.
- const loadAction_optional & loadAction () const
  Return a read-only (constant) reference to the element container.
- loadAction_optional & loadAction ()
  Return a read-write reference to the element container.
- void loadAction (const loadAction_type &x)
  Set the element value.
- void loadAction (const loadAction_optional &x)
  Set the element value.
- void loadAction (::std::unique_ptr< loadAction_type > p)
  Set the element value without copying.

Constructors

- onAttack ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- onAttack (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- onAttack (const onAttack &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual onAttack * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- onAttack & operator= (const onAttack &x)

  Copy assignment operator.
- virtual ~onAttack ()

  Destructor.

16.82.1 Detailed Description

Class corresponding to the onAttack schema type.

16.82.2 Constructor & Destructor Documentation

16.82.2.1 onAttack() [1/2]

```
Common::onAttack::onAttack (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.82.2.2 onAttack() [2/2]

```
Common::onAttack::onAttack (
            const onAttack & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.82.3 Member Function Documentation

16.82.3.1 _clone()

```
onAttack * Common::onAttack::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.82.3.2 loadAction() [1/5]

```
onAttack::loadAction_optional & Common::onAttack::loadAction ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.82.3.3 loadAction() [2/5]

```
const onAttack::loadAction_optional & Common::onAttack::loadAction ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.82.3.4 loadAction() [3/5]

```
void Common::onAttack::loadAction (
            ::std::unique_ptr< loadAction_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.82.3.5 loadAction() [4/5]

```
void Common::onAttack::loadAction (
            const loadAction_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.82.3.6 loadAction() [5/5]

```
void Common::onAttack::loadAction (
            const loadAction_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.82.3.7 operator=()

```
onAttack & Common::onAttack::operator= (
            const onAttack & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.82.3.8 playSound() [1/5]

```
onAttack::playSound_optional & Common::onAttack::playSound ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.82.3.9 playSound() [2/5]

```
const onAttack::playSound_optional & Common::onAttack::playSound ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.82.3.10 playSound() [3/5]

```
void Common::onAttack::playSound (
            ::std::unique_ptr< playSound_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.82.3.11 playSound() [4/5]

```
void Common::onAttack::playSound (
            const playSound_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.82.3.12 playSound() [5/5]

```
void Common::onAttack::playSound (
            const playSound_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

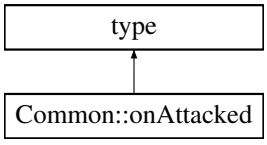
- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.83 Common::onAttacked Class Reference

Class corresponding to the onAttacked schema type.

`#include <common.hxx>`

Inheritance diagram for Common::onAttacked:



playSound

Accessor and modifier functions for the playSound optional element.

- typedef ::xml_schema::uri playSound_type

  Element type.
- typedef ::xsd::cxx::tree::optional< playSound_type > playSound_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< playSound_type, char > playSound_traits

  Element traits type.
- const playSound_optional & playSound () const

  Return a read-only (constant) reference to the element container.
- playSound_optional & playSound ()

  Return a read-write reference to the element container.
- void playSound (const playSound_type &x)

  Set the element value.
- void playSound (const playSound_optional &x)

  Set the element value.
- void playSound (::std::unique_ptr< playSound_type > p)

  Set the element value without copying.

loadAction

Accessor and modifier functions for the loadAction optional element.

- typedef ::xml_schema::uri loadAction_type

  Element type.
- typedef ::xsd::cxx::tree::optional< loadAction_type > loadAction_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< loadAction_type, char > loadAction_traits

  Element traits type.
- const loadAction_optional & loadAction () const

  Return a read-only (constant) reference to the element container.
- loadAction_optional & loadAction ()

  Return a read-write reference to the element container.
- void loadAction (const loadAction_type &x)

  Set the element value.
- void loadAction (const loadAction_optional &x)

  Set the element value.
- void loadAction (::std::unique_ptr< loadAction_type > p)

  Set the element value without copying.

Constructors

- onAttacked ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- onAttacked (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- onAttacked (const onAttacked &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual onAttacked * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- onAttacked & operator= (const onAttacked &x)

  Copy assignment operator.
- virtual ~onAttacked ()

  Destructor.

16.83.1 Detailed Description

Class corresponding to the onAttacked schema type.

16.83.2 Constructor & Destructor Documentation

16.83.2.1 onAttacked() [1/2]

```
Common::onAttacked::onAttacked (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.83.2.2 onAttacked() [2/2]

```
Common::onAttacked::onAttacked (
            const onAttacked & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.83.3 Member Function Documentation

16.83.3.1 _clone()

```
onAttacked * Common::onAttacked::_clone (
             ::xml_schema::flags f = 0,
             ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.83.3.2 loadAction() [1/5]

```
onAttacked::loadAction_optional & Common::onAttacked::loadAction ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.83.3.3 loadAction() [2/5]

```
const onAttacked::loadAction_optional & Common::onAttacked::loadAction ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.83.3.4 loadAction() [3/5]

```
void Common::onAttacked::loadAction (
            ::std::unique_ptr< loadAction_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.83.3.5 loadAction() [4/5]

```
void Common::onAttacked::loadAction (
            const loadAction_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.83.3.6 loadAction() [5/5]

```
void Common::onAttacked::loadAction (
            const loadAction_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.83.3.7 operator=()

```
onAttacked & Common::onAttacked::operator= (
            const onAttacked & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.83.3.8 playSound() [1/5]

```
onAttacked::playSound_optional & Common::onAttacked::playSound ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.83.3.9 playSound() [2/5]

```
const onAttacked::playSound_optional & Common::onAttacked::playSound ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.83.3.10 playSound() [3/5]

```
void Common::onAttacked::playSound (
            ::std::unique_ptr< playSound_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.83.3.11 playSound() [4/5]

```
void Common::onAttacked::playSound (
            const playSound_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.83.3.12 playSound() [5/5]

```
void Common::onAttacked::playSound (
            const playSound_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

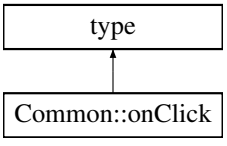
- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.84 Common::onClick Class Reference

Class corresponding to the onClick schema type.

`#include <common.hxx>`

Inheritance diagram for Common::onClick:



playSound

Accessor and modifier functions for the playSound optional element.

- typedef ::xml_schema::uri playSound_type

  Element type.
- typedef ::xsd::cxx::tree::optional< playSound_type > playSound_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< playSound_type, char > playSound_traits

  Element traits type.
- const playSound_optional & playSound () const

  Return a read-only (constant) reference to the element container.
- playSound_optional & playSound ()

  Return a read-write reference to the element container.
- void playSound (const playSound_type &x)

  Set the element value.
- void playSound (const playSound_optional &x)

  Set the element value.
- void playSound (::std::unique_ptr< playSound_type > p)

  Set the element value without copying.

loadScene

Accessor and modifier functions for the loadScene optional element.

- typedef ::xml_schema::uri loadScene_type

  Element type.
- typedef ::xsd::cxx::tree::optional< loadScene_type > loadScene_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< loadScene_type, char > loadScene_traits

  Element traits type.
- const loadScene_optional & loadScene () const

  Return a read-only (constant) reference to the element container.
- loadScene_optional & loadScene ()

  Return a read-write reference to the element container.
- void loadScene (const loadScene_type &x)

  Set the element value.
- void loadScene (const loadScene_optional &x)

  Set the element value.
- void loadScene (::std::unique_ptr< loadScene_type > p)

  Set the element value without copying.

custom

Accessor and modifier functions for the custom optional element.

- typedef ::xml_schema::string custom_type

  Element type.
- typedef ::xsd::cxx::tree::optional< custom_type > custom_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< custom_type, char > custom_traits

  Element traits type.
- const custom_optional & custom () const

  Return a read-only (constant) reference to the element container.
- custom_optional & custom ()

  Return a read-write reference to the element container.
- void custom (const custom_type &x)

  Set the element value.
- void custom (const custom_optional &x)

  Set the element value.
- void custom (::std::unique_ptr< custom_type > p)

  Set the element value without copying.

loadURL

Accessor and modifier functions for the loadURL optional element.

- typedef ::xml_schema::string loadURL_type

  Element type.
- typedef ::xsd::cxx::tree::optional< loadURL_type > loadURL_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< loadURL_type, char > loadURL_traits

  Element traits type.
- const loadURL_optional & loadURL () const

  Return a read-only (constant) reference to the element container.
- loadURL_optional & loadURL ()

  Return a read-write reference to the element container.
- void loadURL (const loadURL_type &x)

  Set the element value.
- void loadURL (const loadURL_optional &x)

  Set the element value.
- void loadURL (::std::unique_ptr< loadURL_type > p)

  Set the element value without copying.

Constructors

- onClick ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- onClick (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- onClick (const onClick &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual onClick * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- onClick & operator= (const onClick &x)

  Copy assignment operator.
- virtual ~onClick ()

  Destructor.

16.84.1 Detailed Description

Class corresponding to the onClick schema type.

16.84.2 Constructor & Destructor Documentation

16.84.2.1 onClick() [1/2]

```
Common::onClick::onClick (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.84.2.2 onClick() [2/2]

```
Common::onClick::onClick (
            const onClick & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.84.3 Member Function Documentation

16.84.3.1 _clone()

```
onClick * Common::onClick::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.84.3.2 custom() [1/5]

```
onClick::custom_optional & Common::onClick::custom ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.84.3.3 custom() [2/5]

```
const onClick::custom_optional & Common::onClick::custom ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.84.3.4 custom() [3/5]

```
void Common::onClick::custom (
            ::std::unique_ptr< custom_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.84.3.5 custom() [4/5]

```
void Common::onClick::custom (
            const custom_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.84.3.6 custom() [5/5]

```
void Common::onClick::custom (
            const custom_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.84.3.7 loadScene() [1/5]

```
onClick::loadScene_optional & Common::onClick::loadScene ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.84.3.8 loadScene() [2/5]

```
const onClick::loadScene_optional & Common::onClick::loadScene ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.84.3.9 loadScene() [3/5]

```
void Common::onClick::loadScene (
            ::std::unique_ptr< loadScene_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.84.3.10 loadScene() [4/5]

```
void Common::onClick::loadScene (
            const loadScene_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.84.3.11 loadScene() [5/5]

```
void Common::onClick::loadScene (
            const loadScene_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.84.3.12 loadURL() [1/5]

```
onClick::loadURL_optional & Common::onClick::loadURL ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.84.3.13 loadURL() [2/5]

```
const onClick::loadURL_optional & Common::onClick::loadURL ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.84.3.14 loadURL() [3/5]

```
void Common::onClick::loadURL (
            ::std::unique_ptr< loadURL_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.84.3.15 loadURL() [4/5]

```
void Common::onClick::loadURL (
            const loadURL_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.84.3.16 loadURL() [5/5]

```
void Common::onClick::loadURL (
            const loadURL_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.84.3.17 operator=()

```
onClick & Common::onClick::operator= (
            const onClick & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.84.3.18 playSound() [1/5]

```
onClick::playSound_optional & Common::onClick::playSound ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.84.3.19 playSound() [2/5]

```
const onClick::playSound_optional & Common::onClick::playSound ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.84.3.20 playSound() [3/5]

```
void Common::onClick::playSound (
            ::std::unique_ptr< playSound_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.84.3.21 playSound() [4/5]

```
void Common::onClick::playSound (
            const playSound_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.84.3.22 playSound() [5/5]

```
void Common::onClick::playSound (
            const playSound_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

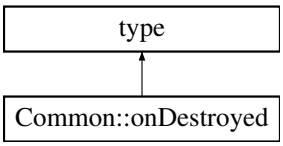
- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.85 Common::onDestroyed Class Reference

Class corresponding to the onDestroyed schema type.

```
#include <common.hxx>
```

Inheritance diagram for Common::onDestroyed:



playSound

Accessor and modifier functions for the playSound optional element.

- typedef ::xml_schema::uri playSound_type

  Element type.
- typedef ::xsd::cxx::tree::optional< playSound_type > playSound_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< playSound_type, char > playSound_traits

  Element traits type.
- const playSound_optional & playSound () const

  Return a read-only (constant) reference to the element container.
- playSound_optional & playSound ()

  Return a read-write reference to the element container.
- void playSound (const playSound_type &x)

  Set the element value.
- void playSound (const playSound_optional &x)

  Set the element value.
- void playSound (::std::unique_ptr< playSound_type > p)

  Set the element value without copying.

loadAction

Accessor and modifier functions for the loadAction optional element.

- typedef ::xml_schema::uri loadAction_type

  Element type.
- typedef ::xsd::cxx::tree::optional< loadAction_type > loadAction_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< loadAction_type, char > loadAction_traits

  Element traits type.
- const loadAction_optional & loadAction () const

  Return a read-only (constant) reference to the element container.
- loadAction_optional & loadAction ()

  Return a read-write reference to the element container.
- void loadAction (const loadAction_type &x)

  Set the element value.
- void loadAction (const loadAction_optional &x)

  Set the element value.
- void loadAction (::std::unique_ptr< loadAction_type > p)

  Set the element value without copying.

Constructors

- onDestroyed ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- onDestroyed (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- onDestroyed (const onDestroyed &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual onDestroyed * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- onDestroyed & operator= (const onDestroyed &x)

  Copy assignment operator.
- virtual ~onDestroyed ()

  Destructor.

16.85.1 Detailed Description

Class corresponding to the onDestroyed schema type.

16.85.2 Constructor & Destructor Documentation

16.85.2.1 onDestroyed() [1/2]

```
Common::onDestroyed::onDestroyed (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.85.2.2 onDestroyed() [2/2]

```
Common::onDestroyed::onDestroyed (
            const onDestroyed & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.85.3 Member Function Documentation

16.85.3.1 _clone()

```
onDestroyed * Common::onDestroyed::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.85.3.2 loadAction() [1/5]

```
onDestroyed::loadAction_optional & Common::onDestroyed::loadAction ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.85.3.3 loadAction() [2/5]

```
const onDestroyed::loadAction_optional & Common::onDestroyed::loadAction ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.85.3.4 loadAction() [3/5]

```
void Common::onDestroyed::loadAction (
            ::std::unique_ptr< loadAction_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.85.3.5 loadAction() [4/5]

```
void Common::onDestroyed::loadAction (
            const loadAction_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.85.3.6 loadAction() [5/5]

```
void Common::onDestroyed::loadAction (
            const loadAction_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.85.3.7 operator=()

```
onDestroyed & Common::onDestroyed::operator= (
            const onDestroyed & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.85.3.8 playSound() [1/5]

```
onDestroyed::playSound_optional & Common::onDestroyed::playSound ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.85.3.9 playSound() [2/5]

```
const onDestroyed::playSound_optional & Common::onDestroyed::playSound ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.85.3.10 playSound() [3/5]

```
void Common::onDestroyed::playSound (
            ::std::unique_ptr< playSound_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.85.3.11 playSound() [4/5]

```
void Common::onDestroyed::playSound (
            const playSound_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.85.3.12 playSound() [5/5]

```
void Common::onDestroyed::playSound (
            const playSound_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

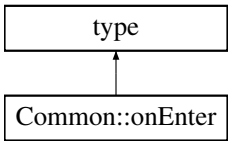
- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.86 Common::onEnter Class Reference

Class corresponding to the onEnter schema type.

```
#include <common.hxx>
```

Inheritance diagram for Common::onEnter:



playSound

Accessor and modifier functions for the playSound optional element.

- typedef ::xml_schema::uri playSound_type

  Element type.
- typedef ::xsd::cxx::tree::optional< playSound_type > playSound_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< playSound_type, char > playSound_traits

  Element traits type.
- const playSound_optional & playSound () const

  Return a read-only (constant) reference to the element container.
- playSound_optional & playSound ()

  Return a read-write reference to the element container.
- void playSound (const playSound_type &x)

  Set the element value.
- void playSound (const playSound_optional &x)

  Set the element value.
- void playSound (::std::unique_ptr< playSound_type > p)

  Set the element value without copying.

loadAction

Accessor and modifier functions for the loadAction optional element.

- typedef ::xml_schema::uri loadAction_type

  Element type.
- typedef ::xsd::cxx::tree::optional< loadAction_type > loadAction_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< loadAction_type, char > loadAction_traits

  Element traits type.
- const loadAction_optional & loadAction () const

  Return a read-only (constant) reference to the element container.
- loadAction_optional & loadAction ()

  Return a read-write reference to the element container.
- void loadAction (const loadAction_type &x)

  Set the element value.
- void loadAction (const loadAction_optional &x)

  Set the element value.
- void loadAction (::std::unique_ptr< loadAction_type > p)

  Set the element value without copying.

Constructors

- onEnter ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- onEnter (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- onEnter (const onEnter &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual onEnter * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- onEnter & operator= (const onEnter &x)

  Copy assignment operator.
- virtual ~onEnter ()

  Destructor.

16.86.1 Detailed Description

Class corresponding to the onEnter schema type.

16.86.2 Constructor & Destructor Documentation

16.86.2.1 onEnter() [1/2]

```
Common::onEnter::onEnter (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.86.2.2 onEnter() [2/2]

```
Common::onEnter::onEnter (
            const onEnter & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.86.3 Member Function Documentation

16.86.3.1 _clone()

```
onEnter * Common::onEnter::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.86.3.2 loadAction() [1/5]

```
onEnter::loadAction_optional & Common::onEnter::loadAction ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.86.3.3 loadAction() [2/5]

```
const onEnter::loadAction_optional & Common::onEnter::loadAction ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.86.3.4 loadAction() [3/5]

```
void Common::onEnter::loadAction (
            ::std::unique_ptr< loadAction_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.86.3.5 loadAction() [4/5]

```
void Common::onEnter::loadAction (
            const loadAction_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.86.3.6 loadAction() [5/5]

```
void Common::onEnter::loadAction (
            const loadAction_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.86.3.7 operator=()

```
onEnter & Common::onEnter::operator= (
            const onEnter & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.86.3.8 playSound() [1/5]

```
onEnter::playSound_optional & Common::onEnter::playSound ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.86.3.9 playSound() [2/5]

```
const onEnter::playSound_optional & Common::onEnter::playSound ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.86.3.10 playSound() [3/5]

```
void Common::onEnter::playSound (
            ::std::unique_ptr< playSound_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.86.3.11 playSound() [4/5]

```
void Common::onEnter::playSound (
            const playSound_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.86.3.12 playSound() [5/5]

```
void Common::onEnter::playSound (
            const playSound_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

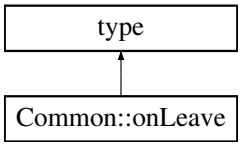
- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.87 Common::onLeave Class Reference

Class corresponding to the onLeave schema type.

```
#include <common.hxx>
```

Inheritance diagram for Common::onLeave:



playSound

Accessor and modifier functions for the playSound optional element.

- typedef ::xml_schema::uri playSound_type

  Element type.
- typedef ::xsd::cxx::tree::optional< playSound_type > playSound_optional

  Element optional container type.

- typedef ::xsd::cxx::tree::traits< playSound_type, char > playSound_traits
  Element traits type.
- const playSound_optional & playSound () const
  Return a read-only (constant) reference to the element container.
- playSound_optional & playSound ()
  Return a read-write reference to the element container.
- void playSound (const playSound_type &x)
  Set the element value.
- void playSound (const playSound_optional &x)
  Set the element value.
- void playSound (::std::unique_ptr< playSound_type > p)
  Set the element value without copying.

loadAction

Accessor and modifier functions for the loadAction optional element.

- typedef ::xml_schema::uri loadAction_type
  Element type.
- typedef ::xsd::cxx::tree::optional< loadAction_type > loadAction_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< loadAction_type, char > loadAction_traits
  Element traits type.
- const loadAction_optional & loadAction () const
  Return a read-only (constant) reference to the element container.
- loadAction_optional & loadAction ()
  Return a read-write reference to the element container.
- void loadAction (const loadAction_type &x)
  Set the element value.
- void loadAction (const loadAction_optional &x)
  Set the element value.
- void loadAction (::std::unique_ptr< loadAction_type > p)
  Set the element value without copying.

Constructors

- onLeave ()
  Create an instance from the ultimate base and initializers for required elements and attributes.
- onLeave (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- onLeave (const onLeave &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual onLeave * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- onLeave & operator= (const onLeave &x)
  Copy assignment operator.
- virtual ~onLeave ()
  Destructor.

16.87.1 Detailed Description

Class corresponding to the onLeave schema type.

16.87.2 Constructor & Destructor Documentation

16.87.2.1 onLeave() [1/2]

```
Common::onLeave::onLeave (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.87.2.2 onLeave() [2/2]

```
Common::onLeave::onLeave (
            const onLeave & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.87.3 Member Function Documentation

16.87.3.1 _clone()

```
onLeave * Common::onLeave::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.87.3.2 loadAction() [1/5]

```
onLeave::loadAction_optional & Common::onLeave::loadAction ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.87.3.3 loadAction() [2/5]

```
const onLeave::loadAction_optional & Common::onLeave::loadAction ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.87.3.4 loadAction() [3/5]

```
void Common::onLeave::loadAction (
            ::std::unique_ptr< loadAction_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.87.3.5 loadAction() [4/5]

```
void Common::onLeave::loadAction (
            const loadAction_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.87.3.6 loadAction() [5/5]

```
void Common::onLeave::loadAction (
            const loadAction_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.87.3.7 operator=()

```
onLeave & Common::onLeave::operator= (
            const onLeave & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.87.3.8 playSound() [1/5]

```
onLeave::playSound_optional & Common::onLeave::playSound ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.87.3.9 playSound() [2/5]

```
const onLeave::playSound_optional & Common::onLeave::playSound ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.87.3.10 playSound() [3/5]

```
void Common::onLeave::playSound (
            ::std::unique_ptr< playSound_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.87.3.11 playSound() [4/5]

```
void Common::onLeave::playSound (
            const playSound_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.87.3.12 playSound() [5/5]

```
void Common::onLeave::playSound (
            const playSound_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.88 org_kde_kwin_server_decoration_listener Struct Reference

Public Attributes

- void(∗ mode)(void ∗data, struct org_kde_kwin_server_decoration ∗org_kde_kwin_server_decoration, uint32_t mode)

16.88.1 Member Data Documentation

16.88.1.1 mode

```
void(* org_kde_kwin_server_decoration_listener::mode) (void *data, struct org_kde_kwin_server↩
_decoration *org_kde_kwin_server_decoration, uint32_t mode)
```

The new decoration mode applied by the server

This event is emitted directly after the decoration is created and represents the base decoration policy by the server. E.g. a server which wants all surfaces to be client-side decorated will send Client, a server which wants server-side decoration will send Server.

The client can request a different mode through the decoration request. The server will acknowledge this by another event with the same mode. So even if a server prefers server-side decoration it's possible to force a client-side decoration.

The server may emit this event at any time. In this case the client can again request a different mode. It's the responsibility of the server to prevent a feedback loop.

Parameters

mode	The decoration mode applied to the surface by the server.

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/org-kde-kwin-server-decoration-manager-client-protocol.h

16.89 org_kde_kwin_server_decoration_manager_listener Struct Reference

Public Attributes

- void(∗ default_mode)(void ∗data, struct org_kde_kwin_server_decoration_manager ∗org_kde_kwin_server_decoration_manager, uint32_t mode)

16.89.1 Member Data Documentation

16.89.1.1 default_mode

```
void(* org_kde_kwin_server_decoration_manager_listener::default_mode) (void *data, struct org_kde_kwin_server_decoration_manager *org_kde_kwin_server_decoration_manager, uint32_t mode)
```

The default mode used on the server

This event is emitted directly after binding the interface. It contains the default mode for the decoration. When a new server decoration object is created this new object will be in the default mode until the first request_mode is requested.

The server may change the default mode at any time.

Parameters

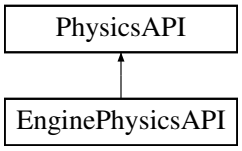
mode	The default decoration mode applied to newly created server decorations.

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/org-kde-kwin-server-decoration-manager-client-protocol.h

16.90 PhysicsAPI Class Reference

Inheritance diagram for PhysicsAPI:



Public Member Functions

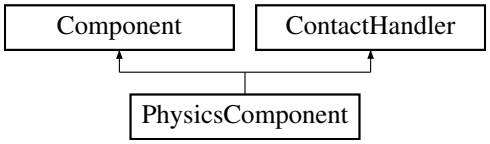
- virtual void **update** (float deltaTime)=0
- virtual BodyId **createBody** (const Box2DBoxData &box2DBoxData) const =0
- virtual BodyId **createBody** (const Box2DCircleData &box2DCircleData) const =0
- virtual BodyId **createBody** (const Box2DPolygonData &box2DPolygonData) const =0
- virtual void **destroyBody** (BodyId bodyId)=0
- virtual RTransform **getRPosition** (BodyId bodyId) const =0
- virtual void **DebugDraw** (const RenderingAPI &renderingApi, SDL_Renderer &renderer)=0
- virtual void **GetVelocity** (Vector2 &velocity, const BodyId bodyId) const =0
- virtual void **setLinearVelocity** (const BodyId bodyId, const Vector2 &vector2) const =0
- virtual void **setFixedRotation** (const BodyId i, bool b) const =0
- virtual void **addForce** (const BodyId i, Vector2) const =0
- virtual void **setAngle** (const BodyId i, float rotation) const =0
- virtual void **destroyBody** (BodyId i) const =0
- virtual PhysicsEngineAdapter & **getPhysicsEngineAdapter** () const =0
- virtual void **setTransform** (unsigned int bodyId, Vector2 pos, float angle) const =0
- virtual void **setEnabled** (BodyId id, bool b) const =0

The documentation for this class was generated from the following file:

- API/Physics/PhysicsAPI.hpp

16.91 PhysicsComponent Class Reference

Inheritance diagram for PhysicsComponent:



Public Member Functions

- const std::string & **getContactHandlerName** ()
- void **destroyBody** ()
- **PhysicsComponent** (EntityId id)
- **PhysicsComponent** (EntityId id, BodyType bodyType, Vector2 position, Vector2 size)
- **PhysicsComponent** (EntityId id, BodyType bodyType, Vector2 position, float radius)
- RTransform **getRTransform** ()
- void **getVelocity** (Vector2 &velocity)
- void **setVelocity** (const Vector2 &velocity)

- void **setFixedRotation** (bool value)
- std::string **name** () const override
- Component * **build** (EntityId entityId, const Components::component *component) override
- void **initialize** (EntityObject &entityParent) override
- void **fixedUpdate** (const float &deltaTime) override
- void **render** () override
- void **update** (const Input &inputSystem) override
- void **startContact** (b2Contact *contact) override
- void **endContact** (b2Contact *contact) override
- void **setAngle** (float angle)
- void **setTransform** (Vector2 pos, float angle)
- void **addForce** (Vector2 dir)
- void **setEnabled** (bool b)

Static Public Member Functions

- static BodyType **StringToBodyType** (const std::string &value)
- static TransformComponent * **setPositionPhysicsResource** (EntityObject *pObject, Components::physicsComponent &component)

Public Attributes

- std::vector< ContactHandler * > **contactHandlers** {}

Additional Inherited Members

The documentation for this class was generated from the following files:

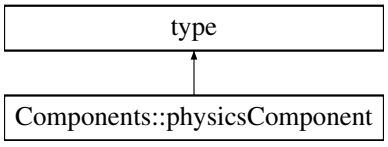
- Game/Components/PhysicsComponent.hpp
- Game/Components/PhysicsComponent.cpp

16.92 Components::physicsComponent Class Reference

Class corresponding to the physicsComponent schema type.

`#include <components.hxx>`

Inheritance diagram for Components::physicsComponent:



position

Accessor and modifier functions for the position required element.

- typedef ::Common::position position_type
  Element type.
- typedef ::xsd::cxx::tree::traits< position_type, char > position_traits
  Element traits type.
- const position_type & position () const
  Return a read-only (constant) reference to the element.
- position_type & position ()
  Return a read-write reference to the element.
- void position (const position_type &x)
  Set the element value.
- void position (::std::unique_ptr< position_type > p)
  Set the element value without copying.

friction

Accessor and modifier functions for the friction required element.

- typedef ::Components::floatCap friction_type
  Element type.
- typedef ::xsd::cxx::tree::traits< friction_type, char > friction_traits
  Element traits type.
- const friction_type & friction () const
  Return a read-only (constant) reference to the element.
- friction_type & friction ()
  Return a read-write reference to the element.
- void friction (const friction_type &x)
  Set the element value.
- void friction (::std::unique_ptr< friction_type > p)
  Set the element value without copying.

bodyType

Accessor and modifier functions for the bodyType required element.

- typedef ::Components::bodyType bodyType_type
  Element type.
- typedef ::xsd::cxx::tree::traits< bodyType_type, char > bodyType_traits
  Element traits type.
- const bodyType_type & bodyType () const
  Return a read-only (constant) reference to the element.
- bodyType_type & bodyType ()
  Return a read-write reference to the element.
- void bodyType (const bodyType_type &x)
  Set the element value.
- void bodyType (::std::unique_ptr< bodyType_type > p)
  Set the element value without copying.

bodyShape

Accessor and modifier functions for the bodyShape required element.

- typedef ::Components::bodyShape bodyShape_type
  Element type.
- typedef ::xsd::cxx::tree::traits< bodyShape_type, char > bodyShape_traits
  Element traits type.
- const bodyShape_type & bodyShape () const
  Return a read-only (constant) reference to the element.
- bodyShape_type & bodyShape ()
  Return a read-write reference to the element.
- void bodyShape (const bodyShape_type &x)
  Set the element value.
- void bodyShape (::std::unique_ptr< bodyShape_type > p)
  Set the element value without copying.

isBullet

Accessor and modifier functions for the isBullet optional element.

- typedef ::xml_schema::boolean isBullet_type
  Element type.
- typedef ::xsd::cxx::tree::optional< isBullet_type > isBullet_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< isBullet_type, char > isBullet_traits
  Element traits type.
- const isBullet_optional & isBullet () const
  Return a read-only (constant) reference to the element container.
- isBullet_optional & isBullet ()
  Return a read-write reference to the element container.
- void isBullet (const isBullet_type &x)
  Set the element value.
- void isBullet (const isBullet_optional &x)
  Set the element value.
- static isBullet_type isBullet_default_value ()
  Return the default value for the element.

isSensor

Accessor and modifier functions for the isSensor optional element.

- typedef ::xml_schema::boolean isSensor_type
  Element type.
- typedef ::xsd::cxx::tree::optional< isSensor_type > isSensor_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< isSensor_type, char > isSensor_traits

Element traits type.

- const isSensor_optional & isSensor () const

  Return a read-only (constant) reference to the element container.
- isSensor_optional & isSensor ()

  Return a read-write reference to the element container.
- void isSensor (const isSensor_type &x)

  Set the element value.
- void isSensor (const isSensor_optional &x)

  Set the element value.
- static isSensor_type isSensor_default_value ()

  Return the default value for the element.

isEnabled

Accessor and modifier functions for the isEnabled optional element.

- typedef ::xml_schema::boolean isEnabled_type

  Element type.
- typedef ::xsd::cxx::tree::optional< isEnabled_type > isEnabled_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< isEnabled_type, char > isEnabled_traits

  Element traits type.
- const isEnabled_optional & isEnabled () const

  Return a read-only (constant) reference to the element container.
- isEnabled_optional & isEnabled ()

  Return a read-write reference to the element container.
- void isEnabled (const isEnabled_type &x)

  Set the element value.
- void isEnabled (const isEnabled_optional &x)

  Set the element value.
- static isEnabled_type isEnabled_default_value ()

  Return the default value for the element.

contactHandler

Accessor and modifier functions for the contactHandler sequence element.

- typedef ::Components::componentName contactHandler_type

  Element type.
- typedef ::xsd::cxx::tree::sequence< contactHandler_type > contactHandler_sequence

  Element sequence container type.
- typedef contactHandler_sequence::iterator contactHandler_iterator

  Element iterator type.
- typedef contactHandler_sequence::const_iterator contactHandler_const_iterator

  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< contactHandler_type, char > contactHandler_traits

  Element traits type.
- const contactHandler_sequence & contactHandler () const

  Return a read-only (constant) reference to the element sequence.
- contactHandler_sequence & contactHandler ()

  Return a read-write reference to the element sequence.
- void contactHandler (const contactHandler_sequence &s)

  Copy elements from a given sequence.

Constructors

- physicsComponent (const position_type &, const friction_type &, const bodyType_type &, const bodyShape_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- physicsComponent (::std::unique_ptr< position_type >, const friction_type &, const bodyType_type &, ↩ ::std::unique_ptr< bodyShape_type >)

  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- physicsComponent (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- physicsComponent (const physicsComponent &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual physicsComponent * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- physicsComponent & operator= (const physicsComponent &x)

  Copy assignment operator.
- virtual ~physicsComponent ()

  Destructor.

16.92.1 Detailed Description

Class corresponding to the physicsComponent schema type.

16.92.2 Constructor & Destructor Documentation

16.92.2.1 physicsComponent() [1/3]

```
Components::physicsComponent::physicsComponent (
            ::std::unique_ptr< position_type > position,
            const friction_type & friction,
            const bodyType_type & bodyType,
            ::std::unique_ptr< bodyShape_type > bodyShape )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.92.2.2 physicsComponent() [2/3]

```
Components::physicsComponent::physicsComponent (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.92.2.3 physicsComponent() [3/3]

```
Components::physicsComponent::physicsComponent (
            const physicsComponent & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.92.3 Member Function Documentation

16.92.3.1 _clone()

```
physicsComponent * Components::physicsComponent::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.92.3.2 bodyShape() [1/4]

```
physicsComponent::bodyShape_type & Components::physicsComponent::bodyShape ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.92.3.3 bodyShape() [2/4]

```
const physicsComponent::bodyShape_type & Components::physicsComponent::bodyShape ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.92.3.4 bodyShape() [3/4]

```
void Components::physicsComponent::bodyShape (
            ::std::unique_ptr< bodyShape_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.92.3.5 bodyShape() [4/4]

```
void Components::physicsComponent::bodyShape (
            const bodyShape_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.92.3.6 bodyType() [1/4]

```
physicsComponent::bodyType_type & Components::physicsComponent::bodyType ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.92.3.7 bodyType() [2/4]

```
const physicsComponent::bodyType_type & Components::physicsComponent::bodyType ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.92.3.8 bodyType() [3/4]

```
void Components::physicsComponent::bodyType (
            ::std::unique_ptr< bodyType_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.92.3.9 bodyType() [4/4]

```
void Components::physicsComponent::bodyType (
            const bodyType_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.92.3.10 contactHandler() [1/3]

```
physicsComponent::contactHandler_sequence & Components::physicsComponent::contactHandler ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.92.3.11 contactHandler() [2/3]

```
const physicsComponent::contactHandler_sequence & Components::physicsComponent::contactHandler
( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.92.3.12 contactHandler() [3/3]

```
void Components::physicsComponent::contactHandler (
            const contactHandler_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

16.92.3.13 friction() [1/4]

```
physicsComponent::friction_type & Components::physicsComponent::friction ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.92.3.14 friction() [2/4]

```
const physicsComponent::friction_type & Components::physicsComponent::friction ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.92.3.15 friction() [3/4]

```
void Components::physicsComponent::friction (
            ::std::unique_ptr< friction_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.92.3.16 friction() [4/4]

```
void Components::physicsComponent::friction (
            const friction_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.92.3.17 isBullet() [1/4]

```
physicsComponent::isBullet_optional & Components::physicsComponent::isBullet ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.92.3.18 isBullet() [2/4]

```
const physicsComponent::isBullet_optional & Components::physicsComponent::isBullet ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.92.3.19 isBullet() [3/4]

```
void Components::physicsComponent::isBullet (
            const isBullet_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.92.3.20 isBullet() [4/4]

```
void Components::physicsComponent::isBullet (
            const isBullet_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.92.3.21 isBullet_default_value()

```
physicsComponent::isBullet_type Components::physicsComponent::isBullet_default_value ( )  [static]
```

Return the default value for the element.

Returns

The element's default value.

16.92.3.22 isEnabled() [1/4]

```
physicsComponent::isEnabled_optional & Components::physicsComponent::isEnabled ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.92.3.23 isEnabled() [2/4]

```
const physicsComponent::isEnabled_optional & Components::physicsComponent::isEnabled ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.92.3.24 isEnabled() [3/4]

```
void Components::physicsComponent::isEnabled (
            const isEnabled_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.92.3.25 isEnabled() [4/4]

```
void Components::physicsComponent::isEnabled (
            const isEnabled_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.92.3.26 isEnabled_default_value()

```
physicsComponent::isEnabled_type Components::physicsComponent::isEnabled_default_value ( )
[static]
```

Return the default value for the element.

Returns

The element's default value.

16.92.3.27 isSensor() [1/4]

```
physicsComponent::isSensor_optional & Components::physicsComponent::isSensor ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.92.3.28 isSensor() [2/4]

```
const physicsComponent::isSensor_optional & Components::physicsComponent::isSensor ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.92.3.29 isSensor() [3/4]

```
void Components::physicsComponent::isSensor (
            const isSensor_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.92.3.30 isSensor() [4/4]

```
void Components::physicsComponent::isSensor (
            const isSensor_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.92.3.31 isSensor_default_value()

```
physicsComponent::isSensor_type Components::physicsComponent::isSensor_default_value ( )  [static]
```

Return the default value for the element.

Returns

The element's default value.

16.92.3.32 operator=()

```
physicsComponent & Components::physicsComponent::operator= (
            const physicsComponent & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.92.3.33 position() [1/4]

```
physicsComponent::position_type & Components::physicsComponent::position ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.92.3.34 position() [2/4]

```
const physicsComponent::position_type & Components::physicsComponent::position ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.92.3.35 position() [3/4]

```
void Components::physicsComponent::position (
            ::std::unique_ptr< position_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.92.3.36 position() [4/4]

```
void Components::physicsComponent::position (
            const position_type & x )
```

Set the element value.

Parameters

x	A new value to set.

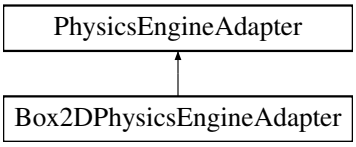
This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.93 PhysicsEngineAdapter Class Reference

Inheritance diagram for PhysicsEngineAdapter:



Public Member Functions

- virtual void **update** (float timeStep)=0
- virtual BodyId **createBody** (const Box2DBoxData &box2dBoxData)=0
- virtual BodyId **createBody** (const Box2DCircleData &box2DCircleData)=0
- virtual BodyId **createBody** (const Box2DPolygonData &box2DPolygonData)=0
- virtual void **referencePositionToBody** (BodyId bodyId, float &x, float &y)=0
- virtual RTransform **getRPosition** (BodyId bodyId)=0
- virtual void **destroyBody** (BodyId BodyID)=0
- virtual void **DebugDraw** (const SDLRenderingAdapter &renderingAdapter, SDL_Renderer &renderer)=0
- virtual void **getVelocity** (Vector2 &velocity, BodyId bodyId) const =0
- virtual void **addForce** (const BodyId i, Vector2 direction) const =0
- virtual void **setLinearVelocity** (const BodyId bodyId, const Vector2 &vector2)=0
- virtual void **setTransform** (unsigned int bodyId, Vector2 pos, float angle) const =0
- virtual void **setFixedRotation** (const BodyId bodyId, bool b)=0
- virtual void **setAngle** (BodyId bodyId, float angle) const =0
- virtual void **setEnabled** (BodyId id, bool b) const =0
- virtual bool **isWorldLocked** () const =0

The documentation for this class was generated from the following file:

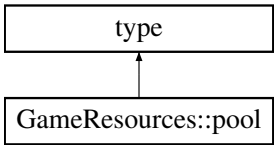
- Engine/Physics/PhysicsEngineAdapter.hpp

16.94 GameResources::pool Class Reference

Class corresponding to the pool schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::pool:



poolName

Accessor and modifier functions for the poolName required element.

- typedef ::xml_schema::string poolName_type
  Element type.
- typedef ::xsd::cxx::tree::traits< poolName_type, char > poolName_traits
  Element traits type.
- const poolName_type & poolName () const
  Return a read-only (constant) reference to the element.
- poolName_type & poolName ()
  Return a read-write reference to the element.
- void poolName (const poolName_type &x)
  Set the element value.
- void poolName (::std::unique_ptr< poolName_type > p)
  Set the element value without copying.

poolPath

Accessor and modifier functions for the poolPath required element.

- typedef ::xml_schema::string poolPath_type
  Element type.
- typedef ::xsd::cxx::tree::traits< poolPath_type, char > poolPath_traits
  Element traits type.
- const poolPath_type & poolPath () const
  Return a read-only (constant) reference to the element.
- poolPath_type & poolPath ()
  Return a read-write reference to the element.
- void poolPath (const poolPath_type &x)
  Set the element value.
- void poolPath (::std::unique_ptr< poolPath_type > p)
  Set the element value without copying.

Constructors

- pool (const poolName_type &, const poolPath_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- pool (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- pool (const pool &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual pool * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- pool & operator= (const pool &x)
  Copy assignment operator.
- virtual ~pool ()
  Destructor.

16.94.1 Detailed Description

Class corresponding to the pool schema type.

16.94.2 Constructor & Destructor Documentation

16.94.2.1 pool() [1/2]

```
GameResources::pool::pool (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.94.2.2 pool() [2/2]

```
GameResources::pool::pool (
            const pool & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.94.3 Member Function Documentation

16.94.3.1 _clone()

```
pool * GameResources::pool::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.94.3.2 operator=()

```
pool & GameResources::pool::operator= (
            const pool & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.94.3.3 poolName() [1/4]

```
pool::poolName_type & GameResources::pool::poolName ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.94.3.4 poolName() [2/4]

```
const pool::poolName_type & GameResources::pool::poolName ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.94.3.5 poolName() [3/4]

```
void GameResources::pool::poolName (
            ::std::unique_ptr< poolName_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.94.3.6 poolName() [4/4]

```
void GameResources::pool::poolName (
            const poolName_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.94.3.7 poolPath() [1/4]

```
pool::poolPath_type & GameResources::pool::poolPath ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.94.3.8 poolPath() [2/4]

```
const pool::poolPath_type & GameResources::pool::poolPath ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.94.3.9 poolPath() [3/4]

```
void GameResources::pool::poolPath (
            ::std::unique_ptr< poolPath_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.94.3.10 poolPath() [4/4]

```
void GameResources::pool::poolPath (
            const poolPath_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.95 Pool Class Reference

Public Member Functions

- const std::string & **getEntityName** () const
- void **initialize** (const std::string &loadList, const std::string &entityName, int startAmount)
- void **resetEntities** ()
- EntityObject * **getEntity** ()

Public Attributes

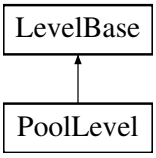
- std::vector< EntityObject * > **entitiesInUse**

The documentation for this class was generated from the following files:

- Game/Object/Pool.hpp
- Game/Object/Pool.cpp

16.96 PoolLevel Class Reference

Inheritance diagram for PoolLevel:



Public Member Functions

- void **initialize** (const std::string &name, const LevelData &data) override
- void **render** () override
- void **update** (const Input &inputSystem) override
- void **fixedUpdate** (float deltaTime) override
- void **clearEntities** () override
- void **addPool** (const std::string &loadList, const std::string &entityName, int startAmount)
- Pool & **getPool** (const std::string &poolName)

Additional Inherited Members

The documentation for this class was generated from the following files:

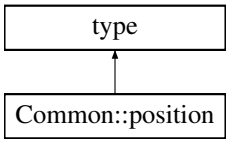
- Game/Scenes/PoolLevel.hpp
- Game/Scenes/PoolLevel.cpp

16.97 Common::position Class Reference

Class corresponding to the position schema type.

`#include <common.hxx>`

Inheritance diagram for Common::position:



x

Accessor and modifier functions for the x required element.

- typedef ::xml_schema::float_ x_type
  Element type.
- typedef ::xsd::cxx::tree::traits< x_type, char > x_traits
  Element traits type.
- const x_type & x () const
  Return a read-only (constant) reference to the element.
- x_type & x ()
  Return a read-write reference to the element.
- void x (const x_type &x)
  Set the element value.

y

Accessor and modifier functions for the y required element.

- typedef ::xml_schema::float_ y_type
  Element type.
- typedef ::xsd::cxx::tree::traits< y_type, char > y_traits
  Element traits type.
- const y_type & y () const
  Return a read-only (constant) reference to the element.
- y_type & y ()
  Return a read-write reference to the element.
- void y (const y_type &x)
  Set the element value.

Constructors

- position (const x_type &, const y_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- position (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- position (const position &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual position * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- position & operator= (const position &x)
  Copy assignment operator.
- virtual ~position ()
  Destructor.

16.97.1 Detailed Description

Class corresponding to the position schema type.

16.97.2 Constructor & Destructor Documentation

16.97.2.1 position() [1/2]

```
Common::position::position (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.97.2.2 position() [2/2]

```
Common::position::position (
            const position & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.97.3 Member Function Documentation

16.97.3.1 _clone()

```
position * Common::position::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.97.3.2 operator=()

```
position & Common::position::operator= (
            const position & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.97.3.3 x() [1/3]

```
position::x_type & Common::position::x ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.97.3.4 x() [2/3]

```
const position::x_type & Common::position::x ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.97.3.5 x() [3/3]

```
void Common::position::x (
            const x_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.97.3.6 y() [1/3]

```
position::y_type & Common::position::y ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.97.3.7 y() [2/3]

```
const position::y_type & Common::position::y ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.97.3.8 y() [3/3]

```
void Common::position::y (
            const y_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

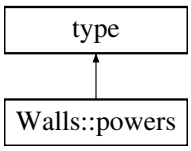
- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.98 Walls::powers Class Reference

Class corresponding to the powers schema type.

```
#include <wall.hxx>
```

Inheritance diagram for Walls::powers:



restoresHP

Accessor and modifier functions for the restoresHP optional element.

- typedef ::xml_schema::int_ restoresHP_type

  Element type.
- typedef ::xsd::cxx::tree::optional< restoresHP_type > restoresHP_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< restoresHP_type, char > restoresHP_traits

  Element traits type.
- const restoresHP_optional & restoresHP () const

  Return a read-only (constant) reference to the element container.
- restoresHP_optional & restoresHP ()

  Return a read-write reference to the element container.
- void restoresHP (const restoresHP_type &x)

  Set the element value.
- void restoresHP (const restoresHP_optional &x)

  Set the element value.

explosionImmune

Accessor and modifier functions for the explosionImmune optional element.

- typedef ::xml_schema::boolean explosionImmune_type
  Element type.
- typedef ::xsd::cxx::tree::optional< explosionImmune_type > explosionImmune_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< explosionImmune_type, char > explosionImmune_traits
  Element traits type.
- const explosionImmune_optional & explosionImmune () const
  Return a read-only (constant) reference to the element container.
- explosionImmune_optional & explosionImmune ()
  Return a read-write reference to the element container.
- void explosionImmune (const explosionImmune_type &x)
  Set the element value.
- void explosionImmune (const explosionImmune_optional &x)
  Set the element value.

Constructors

- powers ()
  Create an instance from the ultimate base and initializers for required elements and attributes.
- powers (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- powers (const powers &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual powers * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- powers & operator= (const powers &x)
  Copy assignment operator.
- virtual ~powers ()
  Destructor.

16.98.1 Detailed Description

Class corresponding to the powers schema type.

16.98.2 Constructor & Destructor Documentation

16.98.2.1 powers() [1/2]

```
Walls::powers::powers (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.98.2.2 powers() [2/2]

```
Walls::powers::powers (
            const powers & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.98.3 Member Function Documentation

16.98.3.1 _clone()

```
powers * Walls::powers::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.98.3.2 explosionImmune() [1/4]

```
powers::explosionImmune_optional & Walls::powers::explosionImmune ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.98.3.3 explosionImmune() [2/4]

```
const powers::explosionImmune_optional & Walls::powers::explosionImmune ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.98.3.4 explosionImmune() [3/4]

```
void Walls::powers::explosionImmune (
            const explosionImmune_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.98.3.5 explosionImmune() [4/4]

```
void Walls::powers::explosionImmune (
            const explosionImmune_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.98.3.6 operator=()

```
powers & Walls::powers::operator= (
            const powers & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.98.3.7 restoresHP() [1/4]

```
powers::restoresHP_optional & Walls::powers::restoresHP ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.98.3.8 restoresHP() [2/4]

```
const powers::restoresHP_optional & Walls::powers::restoresHP ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.98.3.9 restoresHP() [3/4]

```
void Walls::powers::restoresHP (
            const restoresHP_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.98.3.10 restoresHP() [4/4]

```
void Walls::powers::restoresHP (
            const restoresHP_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

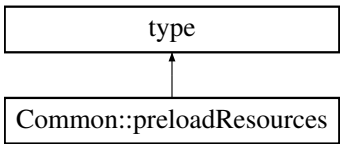
- Resources/XML/Generated/wall.hxx
- Resources/XML/Generated/wall.cxx

16.99 Common::preloadResources Class Reference

Class corresponding to the preloadResources schema type.

```
#include <common.hxx>
```

Inheritance diagram for Common::preloadResources:



resource

Accessor and modifier functions for the resource sequence element.

- typedef ::xml_schema::string resource_type

  Element type.
- typedef ::xsd::cxx::tree::sequence< resource_type > resource_sequence

  Element sequence container type.

- typedef resource_sequence::iterator resource_iterator

  Element iterator type.
- typedef resource_sequence::const_iterator resource_const_iterator

  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< resource_type, char > resource_traits

  Element traits type.
- const resource_sequence & resource () const

  Return a read-only (constant) reference to the element sequence.
- resource_sequence & resource ()

  Return a read-write reference to the element sequence.
- void resource (const resource_sequence &s)

  Copy elements from a given sequence.

Constructors

- preloadResources ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- preloadResources (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- preloadResources (const preloadResources &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual preloadResources * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- preloadResources & operator= (const preloadResources &x)

  Copy assignment operator.
- virtual ~preloadResources ()

  Destructor.

16.99.1 Detailed Description

Class corresponding to the preloadResources schema type.

16.99.2 Constructor & Destructor Documentation

16.99.2.1 preloadResources() [1/2]

```
Common::preloadResources::preloadResources (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.99.2.2 preloadResources() [2/2]

```
Common::preloadResources::preloadResources (
            const preloadResources & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.99.3 Member Function Documentation

16.99.3.1 _clone()

```
preloadResources * Common::preloadResources::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.99.3.2 operator=()

```
preloadResources & Common::preloadResources::operator= (
            const preloadResources & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.99.3.3 resource() [1/3]

```
preloadResources::resource_sequence & Common::preloadResources::resource ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.99.3.4 resource() [2/3]

```
const preloadResources::resource_sequence & Common::preloadResources::resource ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.99.3.5 resource() [3/3]

```
void Common::preloadResources::resource (
            const resource_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

The documentation for this class was generated from the following files:

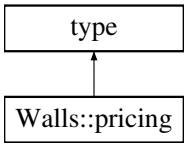
- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.100 Walls::pricing Class Reference

Class corresponding to the pricing schema type.

```
#include <wall.hxx>
```

Inheritance diagram for Walls::pricing:



cost

Accessor and modifier functions for the cost required element.

- typedef ::xml_schema::int_ cost_type

  Element type.
- typedef ::xsd::cxx::tree::traits< cost_type, char > cost_traits

  Element traits type.
- const cost_type & cost () const

  Return a read-only (constant) reference to the element.
- cost_type & cost ()

  Return a read-write reference to the element.
- void cost (const cost_type &x)

  Set the element value.

upgrade

Accessor and modifier functions for the upgrade optional element.

- typedef ::Walls::upgrade upgrade_type
  Element type.
- typedef ::xsd::cxx::tree::optional< upgrade_type > upgrade_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< upgrade_type, char > upgrade_traits
  Element traits type.
- const upgrade_optional & upgrade () const
  Return a read-only (constant) reference to the element container.
- upgrade_optional & upgrade ()
  Return a read-write reference to the element container.
- void upgrade (const upgrade_type &x)
  Set the element value.
- void upgrade (const upgrade_optional &x)
  Set the element value.
- void upgrade (::std::unique_ptr< upgrade_type > p)
  Set the element value without copying.

Constructors

- pricing (const cost_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- pricing (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- pricing (const pricing &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual pricing * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- pricing & operator= (const pricing &x)
  Copy assignment operator.
- virtual ~pricing ()
  Destructor.

16.100.1 Detailed Description

Class corresponding to the pricing schema type.

16.100.2 Constructor & Destructor Documentation

16.100.2.1 pricing() [1/2]

```
Walls::pricing::pricing (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.100.2.2 pricing() [2/2]

```
Walls::pricing::pricing (
            const pricing & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.100.3 Member Function Documentation

16.100.3.1 _clone()

```
pricing * Walls::pricing::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.100.3.2 cost() [1/3]

```
pricing::cost_type & Walls::pricing::cost ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.100.3.3 cost() [2/3]

```
const pricing::cost_type & Walls::pricing::cost ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.100.3.4 cost() [3/3]

```
void Walls::pricing::cost (
            const cost_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.100.3.5 operator=()

```
pricing & Walls::pricing::operator= (
            const pricing & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.100.3.6 upgrade() [1/5]

```
pricing::upgrade_optional & Walls::pricing::upgrade ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.100.3.7 upgrade() [2/5]

```
const pricing::upgrade_optional & Walls::pricing::upgrade ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.100.3.8 upgrade() [3/5]

```
void Walls::pricing::upgrade (
            ::std::unique_ptr< upgrade_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.100.3.9 upgrade() [4/5]

```
void Walls::pricing::upgrade (
            const upgrade_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.100.3.10 upgrade() [5/5]

```
void Walls::pricing::upgrade (
            const upgrade_type & x )
```

Set the element value.

Parameters

x	A new value to set.

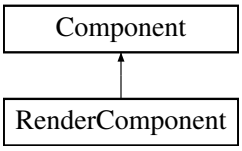
This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/wall.hxx
- Resources/XML/Generated/wall.cxx

16.101 RenderComponent Class Reference

Inheritance diagram for RenderComponent:



Public Member Functions

- void **update** (const Input &inputSystem) override
- void **fixedUpdate** (const float &deltaTime) override
- void setColor (int red, int blue, int green)
- void render () override
- RenderComponent (EntityId id)
- RenderComponent (EntityId id, TransformComponent *transform, const std::string &texturePath, std::string textureId)
- **RenderComponent** (EntityId id, TransformComponent *positionComponent, std::string textureId)
- std::string **name** () const override
- Component * **build** (EntityId entityId, const Components::component *component) override
- void **initialize** (EntityObject &entityParent) override

Additional Inherited Members

16.101.1 Constructor & Destructor Documentation

16.101.1.1 RenderComponent() [1/2]

```
RenderComponent::RenderComponent (
            EntityId id )  [explicit]
```

This is a sample component, this one renders an imahe on the screen.

16.101.1.2 RenderComponent() [2/2]

```
RenderComponent::RenderComponent (
            EntityId id,
            TransformComponent * transform,
            const std::string & texturePath,
            std::string textureId )
```

Instantiates the rendering and passes any needed variables like the engineRenderingApi

Parameters

id	
transform	
texturePath	
textureId	
engineRenderingApi	

16.101.2 Member Function Documentation

16.101.2.1 render()

```
void RenderComponent::render ( )  [override], [virtual]
```

Sample rendering function, the _components are pretty much a poc.

Implements Component.

16.101.2.2 setColor()

```
void RenderComponent::setColor (
            int red,
            int blue,
            int green )
```

Parameters

red	
blue	
green	

The documentation for this class was generated from the following files:

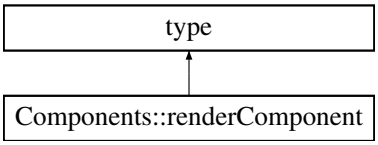
- Game/Components/RenderComponent.hpp
- Game/Components/RenderComponent.cpp

16.102 Components::renderComponent Class Reference

Class corresponding to the renderComponent schema type.

`#include <components.hxx>`

Inheritance diagram for Components::renderComponent:



spriteId

Accessor and modifier functions for the spriteId required element.

- typedef ::xml_schema::string spriteId_type

  Element type.
- typedef ::xsd::cxx::tree::traits< spriteId_type, char > spriteId_traits

  Element traits type.
- const spriteId_type & spriteId () const

  Return a read-only (constant) reference to the element.
- spriteId_type & spriteId ()

  Return a read-write reference to the element.
- void spriteId (const spriteId_type &x)

  Set the element value.
- void spriteId (::std::unique_ptr< spriteId_type > p)

  Set the element value without copying.

spritePath

Accessor and modifier functions for the spritePath required element.

- typedef ::xml_schema::string spritePath_type
  Element type.
- typedef ::xsd::cxx::tree::traits< spritePath_type, char > spritePath_traits
  Element traits type.
- const spritePath_type & spritePath () const
  Return a read-only (constant) reference to the element.
- spritePath_type & spritePath ()
  Return a read-write reference to the element.
- void spritePath (const spritePath_type &x)
  Set the element value.
- void spritePath (::std::unique_ptr< spritePath_type > p)
  Set the element value without copying.

width

Accessor and modifier functions for the width required element.

- typedef ::xml_schema::float_ width_type
  Element type.
- typedef ::xsd::cxx::tree::traits< width_type, char > width_traits
  Element traits type.
- const width_type & width () const
  Return a read-only (constant) reference to the element.
- width_type & width ()
  Return a read-write reference to the element.
- void width (const width_type &x)
  Set the element value.

height

Accessor and modifier functions for the height required element.

- typedef ::xml_schema::float_ height_type
  Element type.
- typedef ::xsd::cxx::tree::traits< height_type, char > height_traits
  Element traits type.
- const height_type & height () const
  Return a read-only (constant) reference to the element.
- height_type & height ()
  Return a read-write reference to the element.
- void height (const height_type &x)
  Set the element value.

Constructors

- renderComponent (const spriteId_type &, const spritePath_type &, const width_type &, const height_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- renderComponent (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- renderComponent (const renderComponent &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual renderComponent * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- renderComponent & operator= (const renderComponent &x)

  Copy assignment operator.
- virtual ~renderComponent ()

  Destructor.

16.102.1 Detailed Description

Class corresponding to the renderComponent schema type.

16.102.2 Constructor & Destructor Documentation

16.102.2.1 renderComponent() [1/2]

```
Components::renderComponent::renderComponent (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.102.2.2 renderComponent() [2/2]

```
Components::renderComponent::renderComponent (
            const renderComponent & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.102.3 Member Function Documentation

16.102.3.1 _clone()

```
renderComponent * Components::renderComponent::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.102.3.2 height() [1/3]

```
renderComponent::height_type & Components::renderComponent::height ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.102.3.3 height() [2/3]

```
const renderComponent::height_type & Components::renderComponent::height ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.102.3.4 height() [3/3]

```
void Components::renderComponent::height (
            const height_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.102.3.5 operator=()

```
renderComponent & Components::renderComponent::operator= (
            const renderComponent & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.102.3.6 spriteId() [1/4]

```
renderComponent::spriteId_type & Components::renderComponent::spriteId ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.102.3.7 spriteId() [2/4]

```
const renderComponent::spriteId_type & Components::renderComponent::spriteId ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.102.3.8 spriteId() [3/4]

```
void Components::renderComponent::spriteId (
            ::std::unique_ptr< spriteId_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.102.3.9 spriteId() [4/4]

```
void Components::renderComponent::spriteId (
            const spriteId_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.102.3.10 spritePath() [1/4]

```
renderComponent::spritePath_type & Components::renderComponent::spritePath ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.102.3.11 spritePath() [2/4]

```
const renderComponent::spritePath_type & Components::renderComponent::spritePath ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.102.3.12 spritePath() [3/4]

```
void Components::renderComponent::spritePath (
            ::std::unique_ptr< spritePath_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.102.3.13 spritePath() [4/4]

```
void Components::renderComponent::spritePath (
            const spritePath_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.102.3.14 width() [1/3]

```
renderComponent::width_type & Components::renderComponent::width ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.102.3.15 width() [2/3]

```
const renderComponent::width_type & Components::renderComponent::width ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.102.3.16 width() [3/3]

```
void Components::renderComponent::width (
            const width_type & x )
```

Set the element value.

Parameters

x	A new value to set.

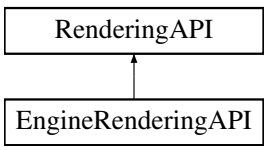
This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.103 RenderingAPI Class Reference

Inheritance diagram for RenderingAPI:



Public Member Functions

- virtual void **drawTexture** (const std::string &textureId, float x, float y, float width, float height, double scale, double r) const =0
- virtual Spritesheet * **createSpriteSheet** (std::string path, std::string spriteSheetId, int width, int height) const =0
- virtual bool **loadTexture** (const std::string &path, const std::string &textureId)=0

- virtual void **drawRectangle** (Vector2 &position, float width, float height, std::string &color, float opacity=1) const =0
- virtual void **createText** (const std::string &fontPath, const std::string &text, const int fontSize, const std::string &hex, const std::string &textureId) const =0
- virtual void **drawBackground** (std::string &hex, float alpha) const =0
- virtual TMXLevel * **loadTMX** (const LevelData &levelData, PhysicsEngineAdapter &physicsEngine↩ Adapter)=0
- virtual void **drawLine** (Vector2 &a, Vector2 &b) const =0
- virtual void **render** () const =0
- virtual const SDLRenderingAdapter & **GetRendererAdapter** () const =0

The documentation for this class was generated from the following file:

- API/Rendering/RenderingAPI.hpp

16.104 ResourceManager Class Reference

Public Member Functions

- void loadRequiredResources (const std::vector< std::string > &resources)
- void **loadResource** (const std::string &resource)
- void **unloadResource** (std::string resource)

Static Public Member Functions

- static ResourceManager * **getInstance** ()
- static ResourceManager * **instantiate** (const std::string &resourcePath, bool debug=true)

Public Attributes

- std::string **_currentLevel**
- bool **quitLevel** = false
- bool **inMenu** = false

Protected Member Functions

- **ResourceManager** (const std::string &resourcePath, bool debug=true)

16.104.1 Member Function Documentation

16.104.1.1 loadRequiredResources()

```
void ResourceManager::loadRequiredResources (
            const std::vector< std::string > & resources )
```

Loads all required resources and unloads non-required ones.

Parameters

resources	

The documentation for this class was generated from the following files:

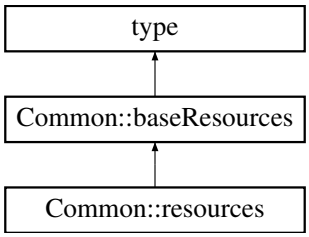
- Engine/Managers/ResourceManager.hpp
- Engine/Managers/ResourceManager.cpp

16.105 Common::resources Class Reference

Class corresponding to the resources schema type.

`#include <common.hxx>`

Inheritance diagram for Common::resources:



Constructors

- resources (const default_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- resources (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- resources (const resources &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual resources * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- virtual ~resources ()

  Destructor.

Additional Inherited Members

16.105.1 Detailed Description

Class corresponding to the resources schema type.

16.105.2 Constructor & Destructor Documentation

16.105.2.1 resources() [1/2]

```
Common::resources::resources (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.105.2.2 resources() [2/2]

```
Common::resources::resources (
            const resources & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.105.3 Member Function Documentation

16.105.3.1 _clone()

```
resources * Common::resources::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

Reimplemented from Common::baseResources.

The documentation for this class was generated from the following files:

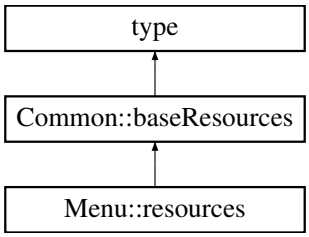
- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.106 Menu::resources Class Reference

Class corresponding to the resources schema type.

`#include <menu.hxx>`

Inheritance diagram for Menu::resources:



hover

Accessor and modifier functions for the hover optional element.

- typedef ::xml_schema::string hover_type
  Element type.
- typedef ::xsd::cxx::tree::optional< hover_type > hover_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< hover_type, char > hover_traits
  Element traits type.
- const hover_optional & hover () const
  Return a read-only (constant) reference to the element container.
- hover_optional & hover ()
  Return a read-write reference to the element container.
- void hover (const hover_type &x)
  Set the element value.
- void hover (const hover_optional &x)
  Set the element value.
- void hover (::std::unique_ptr< hover_type > p)
  Set the element value without copying.

click

Accessor and modifier functions for the click optional element.

- typedef ::xml_schema::string click_type
  Element type.
- typedef ::xsd::cxx::tree::optional< click_type > click_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< click_type, char > click_traits
  Element traits type.
- const click_optional & click () const
  Return a read-only (constant) reference to the element container.
- click_optional & click ()
  Return a read-write reference to the element container.
- void click (const click_type &x)
  Set the element value.
- void click (const click_optional &x)
  Set the element value.
- void click (::std::unique_ptr< click_type > p)
  Set the element value without copying.

Constructors

- resources (const default_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- resources (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- resources (const resources &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual resources * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- resources & operator= (const resources &x)
  Copy assignment operator.
- virtual ~resources ()
  Destructor.

Additional Inherited Members

16.106.1 Detailed Description

Class corresponding to the resources schema type.

16.106.2 Constructor & Destructor Documentation

16.106.2.1 resources() [1/2]

```
Menu::resources::resources (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.106.2.2 resources() [2/2]

```
Menu::resources::resources (
            const resources & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.106.3 Member Function Documentation

16.106.3.1 _clone()

```
resources * Menu::resources::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

Reimplemented from Common::baseResources.

16.106.3.2 click() [1/5]

```
resources::click_optional & Menu::resources::click ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.106.3.3 click() [2/5]

```
const resources::click_optional & Menu::resources::click ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.106.3.4 click() [3/5]

```
void Menu::resources::click (
            ::std::unique_ptr< click_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.106.3.5 click() [4/5]

```
void Menu::resources::click (
            const click_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.106.3.6 click() [5/5]

```
void Menu::resources::click (
            const click_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.106.3.7 hover() [1/5]

```
resources::hover_optional & Menu::resources::hover ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.106.3.8 hover() [2/5]

```
const resources::hover_optional & Menu::resources::hover ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.106.3.9 hover() [3/5]

```
void Menu::resources::hover (
            ::std::unique_ptr< hover_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.106.3.10 hover() [4/5]

```
void Menu::resources::hover (
            const hover_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.106.3.11 hover() [5/5]

```
void Menu::resources::hover (
            const hover_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.106.3.12 operator=()

```
resources & Menu::resources::operator= (
            const resources & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

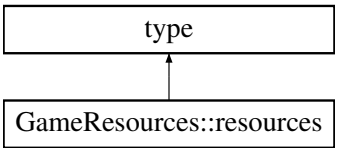
- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.107 GameResources::resources Class Reference

Class corresponding to the resources schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::resources:



basePath

Accessor and modifier functions for the basePath required element.

- typedef ::xml_schema::string basePath_type

  Element type.
- typedef ::xsd::cxx::tree::traits< basePath_type, char > basePath_traits

  Element traits type.
- const basePath_type & basePath () const

  Return a read-only (constant) reference to the element.
- basePath_type & basePath ()

  Return a read-write reference to the element.
- void basePath (const basePath_type &x)

  Set the element value.
- void basePath (::std::unique_ptr< basePath_type > p)

  Set the element value without copying.

textures

Accessor and modifier functions for the textures required element.

- typedef ::GameResources::textures textures_type

  Element type.
- typedef ::xsd::cxx::tree::traits< textures_type, char > textures_traits

  Element traits type.
- const textures_type & textures () const

  Return a read-only (constant) reference to the element.
- textures_type & textures ()

  Return a read-write reference to the element.
- void textures (const textures_type &x)

  Set the element value.
- void textures (::std::unique_ptr< textures_type > p)

  Set the element value without copying.

sprites

Accessor and modifier functions for the sprites required element.

- typedef ::GameResources::sprites sprites_type
  Element type.
- typedef ::xsd::cxx::tree::traits< sprites_type, char > sprites_traits
  Element traits type.
- const sprites_type & sprites () const
  Return a read-only (constant) reference to the element.
- sprites_type & sprites ()
  Return a read-write reference to the element.
- void sprites (const sprites_type &x)
  Set the element value.
- void sprites (::std::unique_ptr< sprites_type > p)
  Set the element value without copying.

sounds

Accessor and modifier functions for the sounds required element.

- typedef ::GameResources::sounds sounds_type
  Element type.
- typedef ::xsd::cxx::tree::traits< sounds_type, char > sounds_traits
  Element traits type.
- const sounds_type & sounds () const
  Return a read-only (constant) reference to the element.
- sounds_type & sounds ()
  Return a read-write reference to the element.
- void sounds (const sounds_type &x)
  Set the element value.
- void sounds (::std::unique_ptr< sounds_type > p)
  Set the element value without copying.

music

Accessor and modifier functions for the music required element.

- typedef ::GameResources::music music_type
  Element type.
- typedef ::xsd::cxx::tree::traits< music_type, char > music_traits
  Element traits type.
- const music_type & music () const
  Return a read-only (constant) reference to the element.
- music_type & music ()
  Return a read-write reference to the element.
- void music (const music_type &x)
  Set the element value.
- void music (::std::unique_ptr< music_type > p)
  Set the element value without copying.

scenes

Accessor and modifier functions for the scenes required element.

- typedef ::GameResources::scenes scenes_type
  Element type.
- typedef ::xsd::cxx::tree::traits< scenes_type, char > scenes_traits
  Element traits type.
- const scenes_type & scenes () const
  Return a read-only (constant) reference to the element.
- scenes_type & scenes ()
  Return a read-write reference to the element.
- void scenes (const scenes_type &x)
  Set the element value.
- void scenes (::std::unique_ptr< scenes_type > p)
  Set the element value without copying.

levels

Accessor and modifier functions for the levels required element.

- typedef ::GameResources::levels levels_type
  Element type.
- typedef ::xsd::cxx::tree::traits< levels_type, char > levels_traits
  Element traits type.
- const levels_type & levels () const
  Return a read-only (constant) reference to the element.
- levels_type & levels ()
  Return a read-write reference to the element.
- void levels (const levels_type &x)
  Set the element value.
- void levels (::std::unique_ptr< levels_type > p)
  Set the element value without copying.

objectLists

Accessor and modifier functions for the objectLists required element.

- typedef ::GameResources::objectLists objectLists_type
  Element type.
- typedef ::xsd::cxx::tree::traits< objectLists_type, char > objectLists_traits
  Element traits type.
- const objectLists_type & objectLists () const
  Return a read-only (constant) reference to the element.
- objectLists_type & objectLists ()
  Return a read-write reference to the element.
- void objectLists (const objectLists_type &x)
  Set the element value.
- void objectLists (::std::unique_ptr< objectLists_type > p)
  Set the element value without copying.

Constructors

- resources (const basePath_type &, const textures_type &, const sprites_type &, const sounds_type &, const music_type &, const scenes_type &, const levels_type &, const objectLists_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- resources (const basePath_type &, ::std::unique_ptr< textures_type >, ::std::unique_ptr< sprites_type >, ::std::unique_ptr< sounds_type >, ::std::unique_ptr< music_type >, ::std::unique_ptr< scenes_type >, ↩ ::std::unique_ptr< levels_type >, ::std::unique_ptr< objectLists_type >)

  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- resources (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- resources (const resources &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual resources * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- resources & operator= (const resources &x)

  Copy assignment operator.
- virtual ~resources ()

  Destructor.

16.107.1 Detailed Description

Class corresponding to the resources schema type.

16.107.2 Constructor & Destructor Documentation

16.107.2.1 resources() [1/3]

```
GameResources::resources::resources (
            const basePath_type & basePath,
            ::std::unique_ptr< textures_type > textures,
            ::std::unique_ptr< sprites_type > sprites,
            ::std::unique_ptr< sounds_type > sounds,
            ::std::unique_ptr< music_type > music,
            ::std::unique_ptr< scenes_type > scenes,
            ::std::unique_ptr< levels_type > levels,
            ::std::unique_ptr< objectLists_type > objectLists )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.107.2.2 resources() [2/3]

```
GameResources::resources::resources (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.107.2.3 resources() [3/3]

```
GameResources::resources::resources (
            const resources & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.107.3 Member Function Documentation

16.107.3.1 _clone()

```
resources * GameResources::resources::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.107.3.2 basePath() [1/4]

```
resources::basePath_type & GameResources::resources::basePath ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.107.3.3 basePath() [2/4]

```
const resources::basePath_type & GameResources::resources::basePath ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.107.3.4 basePath() [3/4]

```
void GameResources::resources::basePath (
            ::std::unique_ptr< basePath_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.107.3.5 basePath() [4/4]

```
void GameResources::resources::basePath (
            const basePath_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.107.3.6 levels() [1/4]

```
resources::levels_type & GameResources::resources::levels ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.107.3.7 levels() [2/4]

```
const resources::levels_type & GameResources::resources::levels ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.107.3.8 levels() [3/4]

```
void GameResources::resources::levels (
            ::std::unique_ptr< levels_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.107.3.9 levels() [4/4]

```
void GameResources::resources::levels (
            const levels_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.107.3.10 music() [1/4]

```
resources::music_type & GameResources::resources::music ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.107.3.11 music() [2/4]

```
const resources::music_type & GameResources::resources::music ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.107.3.12 music() [3/4]

```
void GameResources::resources::music (
            ::std::unique_ptr< music_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.107.3.13 music() [4/4]

```
void GameResources::resources::music (
            const music_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.107.3.14 objectLists() [1/4]

```
resources::objectLists_type & GameResources::resources::objectLists ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.107.3.15 objectLists() [2/4]

```
const resources::objectLists_type & GameResources::resources::objectLists ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.107.3.16 objectLists() [3/4]

```
void GameResources::resources::objectLists (
            ::std::unique_ptr< objectLists_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.107.3.17 objectLists() [4/4]

```
void GameResources::resources::objectLists (
```

```
            const objectLists_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.107.3.18 operator=()

```
resources & GameResources::resources::operator= (
            const resources & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.107.3.19 scenes() [1/4]

```
resources::scenes_type & GameResources::resources::scenes ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.107.3.20 scenes() [2/4]

```
const resources::scenes_type & GameResources::resources::scenes ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.107.3.21 scenes() [3/4]

```
void GameResources::resources::scenes (
            ::std::unique_ptr< scenes_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.107.3.22 scenes() [4/4]

```
void GameResources::resources::scenes (
            const scenes_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.107.3.23 sounds() [1/4]

```
resources::sounds_type & GameResources::resources::sounds ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.107.3.24 sounds() [2/4]

```
const resources::sounds_type & GameResources::resources::sounds ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.107.3.25 sounds() [3/4]

```
void GameResources::resources::sounds (
            ::std::unique_ptr< sounds_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.107.3.26 sounds() [4/4]

```
void GameResources::resources::sounds (
            const sounds_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.107.3.27 sprites() [1/4]

```
resources::sprites_type & GameResources::resources::sprites ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.107.3.28 sprites() [2/4]

```
const resources::sprites_type & GameResources::resources::sprites ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.107.3.29 sprites() [3/4]

```
void GameResources::resources::sprites (
            ::std::unique_ptr< sprites_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.107.3.30 sprites() [4/4]

```
void GameResources::resources::sprites (
            const sprites_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.107.3.31 textures() [1/4]

```
resources::textures_type & GameResources::resources::textures ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.107.3.32 textures() [2/4]

```
const resources::textures_type & GameResources::resources::textures ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.107.3.33 textures() [3/4]

```
void GameResources::resources::textures (
            ::std::unique_ptr< textures_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.107.3.34 textures() [4/4]

```
void GameResources::resources::textures (
            const textures_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.108 RTransform Struct Reference

Public Member Functions

- **RTransform** (const float &x, const float &y, float rotation)

Public Attributes

- const float & **X**
- const float & **Y**
- float **rotation**

The documentation for this struct was generated from the following file:

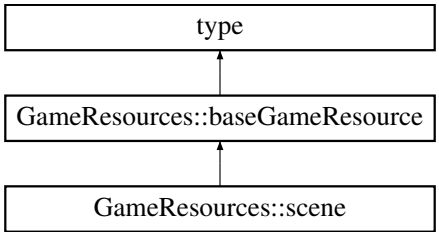
- API/RTransform.hpp

16.109 GameResources::scene Class Reference

Class corresponding to the scene schema type.

```
#include <resources.hxx>
```

Inheritance diagram for GameResources::scene:



Constructors

- scene (const name_type &, const path_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- scene (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- scene (const scene &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual scene * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- virtual ~scene ()

  Destructor.

Additional Inherited Members

16.109.1 Detailed Description

Class corresponding to the scene schema type.

16.109.2 Constructor & Destructor Documentation

16.109.2.1 scene() [1/2]

```
GameResources::scene::scene (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.109.2.2 scene() [2/2]

```
GameResources::scene::scene (
            const scene & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.109.3 Member Function Documentation

16.109.3.1 _clone()

```
scene * GameResources::scene::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

Reimplemented from GameResources::baseGameResource.

The documentation for this class was generated from the following files:

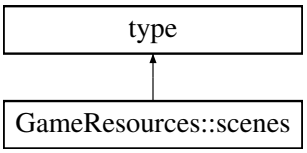
- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.110 GameResources::scenes Class Reference

Class corresponding to the scenes schema type.

```
#include <resources.hxx>
```

Inheritance diagram for GameResources::scenes:



scene

Accessor and modifier functions for the scene sequence element.

- typedef ::GameResources::scene scene_type

  Element type.
- typedef ::xsd::cxx::tree::sequence< scene_type > scene_sequence

  Element sequence container type.
- typedef scene_sequence::iterator scene_iterator

  Element iterator type.
- typedef scene_sequence::const_iterator scene_const_iterator

  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< scene_type, char > scene_traits

  Element traits type.
- const scene_sequence & scene () const

  Return a read-only (constant) reference to the element sequence.
- scene_sequence & scene ()

  Return a read-write reference to the element sequence.
- void scene (const scene_sequence &s)

  Copy elements from a given sequence.

Constructors

- scenes ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- scenes (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- scenes (const scenes &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual scenes * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- scenes & operator= (const scenes &x)

  Copy assignment operator.
- virtual ~scenes ()

  Destructor.

16.110.1 Detailed Description

Class corresponding to the scenes schema type.

16.110.2 Constructor & Destructor Documentation

16.110.2.1 scenes() [1/2]

```
GameResources::scenes::scenes (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.110.2.2 scenes() [2/2]

```
GameResources::scenes::scenes (
            const scenes & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.110.3 Member Function Documentation

16.110.3.1 _clone()

```
scenes * GameResources::scenes::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.110.3.2 operator=()

```
scenes & GameResources::scenes::operator= (
            const scenes & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.110.3.3 scene() [1/3]

```
scenes::scene_sequence & GameResources::scenes::scene ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.110.3.4 scene() [2/3]

```
const scenes::scene_sequence & GameResources::scenes::scene ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.110.3.5 scene() [3/3]

```
void GameResources::scenes::scene (
            const scene_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

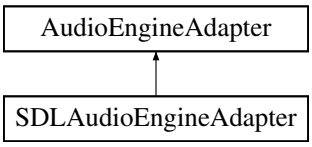
For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.111 SDLAudioEngineAdapter Class Reference

Inheritance diagram for SDLAudioEngineAdapter:



Public Member Functions

- SDLAudioEngineAdapter ()
- **SDLAudioEngineAdapter** (const SDLAudioEngineAdapter &other)=default
- **SDLAudioEngineAdapter** (SDLAudioEngineAdapter &&other) noexcept=default
- SDLAudioEngineAdapter & **operator=** (const SDLAudioEngineAdapter &other)
- SDLAudioEngineAdapter & **operator=** (SDLAudioEngineAdapter &&other) noexcept=default
- std::vector< std::string > getAudioNames () override
- void loadInMemory (const std::string &path, const std::string &name, AudioType type) override
- void playFromPath (const std::string &path, AudioType &type) override
- void playFromMemory (const std::string &name) override
- void changeMasterVolume (int volume) override
- void changeChannelVolume (int channel, int volume) override
- void changeMusicVolume (int volume) override
- int getChannelsAverageVolume () override
- int getChannelVolume (int channel) override
- int getMusicVolume () override
- void stopAudio () override
- void stopMusic () override
- void stopSound (int channel) override
- void stopSounds () override
- void toggleMusic () override
- void toggleSound (int channel) override
- void toggleSounds () override

Static Public Member Functions

- static SDLAudioEngineAdapter & **getInstance** ()

16.111.1 Constructor & Destructor Documentation

16.111.1.1 SDLAudioEngineAdapter()

```
SDLAudioEngineAdapter::SDLAudioEngineAdapter ( )
```

A class that holds all the needed functions to load in and play music

You can either choose to load a sound file directly from a file path or trough memory by adding it to a dictionary

Some functions are declared static for easier sound managing and optimisation. Only loading and playing of a new file requires an instance of the object Constructor initializes the Mix_OpenAudio en Mix_init once so it can be used later.

16.111.2 Member Function Documentation

16.111.2.1 changeChannelVolume()

```
void SDLAudioEngineAdapter::changeChannelVolume (
            int channel,
            int volume )  [override], [virtual]
```

Manages the volume of the specified sound channel

Template Parameters

int	

Parameters

channel	

Template Parameters

int	

Parameters

volume	

Implements AudioEngineAdapter.

16.111.2.2 changeMasterVolume()

```
void SDLAudioEngineAdapter::changeMasterVolume (
            int volume )  [override], [virtual]
```

Manages the volume of the music and all the sound channels

Template Parameters

int	

Parameters

volume	

Implements AudioEngineAdapter.

16.111.2.3 changeMusicVolume()

```
void SDLAudioEngineAdapter::changeMusicVolume (
            int volume ) [override], [virtual]
```

Manages the volume of the music channel

Template Parameters

int	

Parameters

volume	

Implements AudioEngineAdapter.

16.111.2.4 getAudioNames()

```
std::vector< std::string > SDLAudioEngineAdapter::getAudioNames ( ) [override], [virtual]
```

Returns a vector of key strings that are loaded in memory

Returns

std::vector<std::string>

Implements AudioEngineAdapter.

16.111.2.5 getChannelsAverageVolume()

```
int SDLAudioEngineAdapter::getChannelsAverageVolume ( ) [override], [virtual]
```

Returns the average volume of all the sound channels except the music channel

Returns

int

Implements AudioEngineAdapter.

16.111.2.6 getChannelVolume()

```
int SDLAudioEngineAdapter::getChannelVolume (
            int channel )  [override], [virtual]
```

Returns volume of the specified channel

Returns

int

Implements AudioEngineAdapter.

16.111.2.7 getMusicVolume()

```
int SDLAudioEngineAdapter::getMusicVolume ( )  [override], [virtual]
```

Returns the volume of the music channel

Returns

int

Implements AudioEngineAdapter.

16.111.2.8 loadInMemory()

```
void SDLAudioEngineAdapter::loadInMemory (
            const std::string & path,
            const std::string & name,
            AudioType type )  [override], [virtual]
```

Uses the given path and type to add the file in the correct map This is used for sounds that are common in the game (Footstep, gunshot, etc..)

Template Parameters

std::string&	

Parameters

path	

Template Parameters

AudioType&	

Parameters

type	

Implements AudioEngineAdapter.

16.111.2.9 playFromMemory()

```
void SDLAudioEngineAdapter::playFromMemory (
            const std::string & name )  [override], [virtual]
```

plays the file that is connected to the given name (string key)

Template Parameters

std::string&	

Parameters

name	

Implements AudioEngineAdapter.

16.111.2.10 playFromPath()

```
void SDLAudioEngineAdapter::playFromPath (
            const std::string & path,
            AudioType & type )  [override], [virtual]
```

Immediately plays the file from the given path. This is used for sounds that are uncommon and are only used in specific cases

Template Parameters

std::string&	

Parameters

path	

Template Parameters

AudioType&	

Parameters

type	

Implements AudioEngineAdapter.

16.111.2.11 stopAudio()

```
void SDLAudioEngineAdapter::stopAudio ( )  [override], [virtual]
```

Stops all the audio that is currently playing

Implements AudioEngineAdapter.

16.111.2.12 stopMusic()

```
void SDLAudioEngineAdapter::stopMusic ( )  [override], [virtual]
```

Stops the music that is currently playing

Implements AudioEngineAdapter.

16.111.2.13 stopSound()

```
void SDLAudioEngineAdapter::stopSound (
            int channel )  [override], [virtual]
```

Stops the sound played in the specified channel

Template Parameters

int	

Parameters

channel	

Implements AudioEngineAdapter.

16.111.2.14 stopSounds()

```
void SDLAudioEngineAdapter::stopSounds ( )  [override], [virtual]
```

Stops all the channel sounds excluding the music channel

Implements AudioEngineAdapter.

16.111.2.15 toggleMusic()

```
void SDLAudioEngineAdapter::toggleMusic ( )  [override], [virtual]
```

Pauses/resumes the music channel

Implements AudioEngineAdapter.

16.111.2.16 toggleSound()

```
void SDLAudioEngineAdapter::toggleSound (
            int channel )  [override], [virtual]
```

Pauses/resumes the specified sound channel

Template Parameters

int	

Parameters

channel	

Implements AudioEngineAdapter.

16.111.2.17 toggleSounds()

```
void SDLAudioEngineAdapter::toggleSounds ( )  [override], [virtual]
```

Pauses/resumes all the sound channels

Implements AudioEngineAdapter.

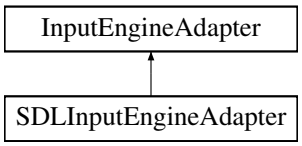
The documentation for this class was generated from the following files:

- Engine/Audio/Adapter/SDLAudioEngineAdapter.hpp
- Engine/Audio/Adapter/SDLAudioEngineAdapter.cpp

16.112 SDLInputEngineAdapter Class Reference

`#include <SDLInputEngineAdapter.hpp>`

Inheritance diagram for SDLInputEngineAdapter:



Public Member Functions

- Event< Input > & **getInputEvent** () override
- Input getInput () override
- Input getKeyInput (SDL_Keycode input, Uint32 type=SDL_KEYDOWN) const override
- Input getMouseInput (SDL_Event input) override
- Input getControllerInput (SDL_Event input) const override
- Input getControllerMotionInput (SDL_Event input) const override
- void **getMousePosition** (int &x, int &y) const override

16.112.1 Detailed Description

SDL Compatible Input Adapter for user input.

16.112.2 Member Function Documentation

16.112.2.1 getControllerInput()

```
Input SDLInputEngineAdapter::getControllerInput (
            SDL_Event controllerEvent ) const  [override], [virtual]
```

Handles Keymapping for device 2: CONTROLLER.

Parameters

SDL_Event	controllerEvent - SDL Event that was received.

Returns

Input{device, x, y, keyMap}

Implements InputEngineAdapter.

16.112.2.2 getControllerMotionInput()

```
Input SDLInputEngineAdapter::getControllerMotionInput (
            SDL_Event controllerEvent ) const  [override], [virtual]
```

Handles Axis motion events for the controller. Note: At this time this only logs which axis was moved.

Parameters

SDL_Event	controllerEvent - SDL Event that was received.

Returns

Input{device, x, y, keyMap}

Left Joystick

Left/Right axis

Left direction

Right direction

Up/Down axis

Up direction

Down direction

Right Joystick

Left/Right axis

Left direction

Right direction

Up/Down axis

Up direction

Down direction

Implements InputEngineAdapter.

16.112.2.3 getInput()

```
Input SDLInputEngineAdapter::getInput ( )  [override], [virtual]
```

Function that gets called every iteration from the gameLoop to check for input from the user. Delegates functionality for every device to underlying methods.

Polled events:

- SDL_KEYDOWN, returns Input struct
- SDL_MOUSEBUTTONDOWN, returns Input struct
- SDL_MOUSEBUTTONDOWN, returns Input struct
- SDL_CONTROLLERDEVICEADDED, void
- SDL_CONTROLLERDEVICEREMOVED, void
- SDL_QUIT, returns Input struct

Returns

Input{device, x, y, keyMap}

Implements InputEngineAdapter.

16.112.2.4 getKeyInput()

```
Input SDLInputEngineAdapter::getKeyInput (
            SDL_Keycode keyEvent,
            Uint32 type = SDL_KEYDOWN ) const  [override], [virtual]
```

Handles Keymapping for device 0: KEYBOARD.

Parameters

SDL_Keycode	keyEvent - SDL KeyEvent that was received.

Returns

Input{device, x, y, keyMap}

Implements InputEngineAdapter.

16.112.2.5 getMouseInput()

```
Input SDLInputEngineAdapter::getMouseInput (
            SDL_Event mouseEvent )  [override], [virtual]
```

Handles Keymapping for device 1: MOUSE.

Parameters

SDL_Event	mouseEvent - SDL Event that was received.

Returns

Input{device, x, y, keyMap}

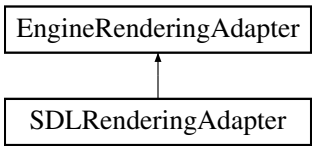
Implements InputEngineAdapter.

The documentation for this class was generated from the following files:

- Engine/Input/Adapter/SDLInputEngineAdapter.hpp
- Engine/Input/Adapter/SDLInputEngineAdapter.cpp

16.113 SDLRenderingAdapter Class Reference

Inheritance diagram for SDLRenderingAdapter:



Public Member Functions

- void **drawTexture** (std::string textureId, float x, float y, float width, float height, double scale, double r, SDL↩ _Renderer ∗renderer, SDL_RendererFlip flip=SDL_FLIP_NONE) override
- Spritesheet ∗ **createSpriteSheet** (const std::string &path, std::string &spriteSheetId, int width, int height) override
- Spritesheet ∗ **createSpriteSheet** (const std::string &path, const std::string &jsonPath, std::string &sprite↩ SheetId, SDL_Renderer ∗renderer) override
- void **drawBox** (const Vector2 ∗vertices, int32 vertexCount, SDL_Renderer ∗renderer) const override
- void **drawLine** (const Vector2 &begin, const Vector2 &end, SDL_Renderer ∗renderer) const override
- void **drawCircle** (const Vector2 &center, const float &radius, SDL_Renderer ∗renderer) const override
- void **drawRectangle** (Vector2 &vector2, float width, float height, const std::string &color, float opacity, SD↩ L_Renderer ∗renderer) const override
- void **createText** (const std::string &fontName, const std::string &text, int fontSize, const std::string &hex, const std::string &textureId, SDL_Renderer ∗renderer) override
- void **drawBackground** (std::string &color, float alpha, SDL_Renderer ∗renderer) override
- void **render** (SDL_Renderer ∗pRenderer) override

Static Public Member Functions

- static TextureManager ∗ **GetTextureManager** ()

The documentation for this class was generated from the following files:

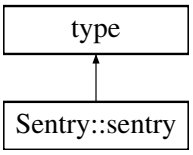
- Engine/Rendering/Adapter/SDLRenderingAdapter.hpp
- Engine/Rendering/Adapter/SDLRenderingAdaper.cpp

16.114 Sentry::sentry Class Reference

Class corresponding to the sentry schema type.

`#include <sentry.hxx>`

Inheritance diagram for Sentry::sentry:



className

Accessor and modifier functions for the className required element.

- typedef ::xml_schema::string className_type
  Element type.
- typedef ::xsd::cxx::tree::traits< className_type, char > className_traits
  Element traits type.
- const className_type & className () const
  Return a read-only (constant) reference to the element.
- className_type & className ()
  Return a read-write reference to the element.
- void className (const className_type &x)
  Set the element value.
- void className (::std::unique_ptr< className_type > p)
  Set the element value without copying.

resources

Accessor and modifier functions for the resources required element.

- typedef ::Common::resources resources_type
  Element type.
- typedef ::xsd::cxx::tree::traits< resources_type, char > resources_traits
  Element traits type.
- const resources_type & resources () const
  Return a read-only (constant) reference to the element.
- resources_type & resources ()
  Return a read-write reference to the element.
- void resources (const resources_type &x)
  Set the element value.
- void resources (::std::unique_ptr< resources_type > p)
  Set the element value without copying.

collider

Accessor and modifier functions for the collider required element.

- typedef ::Common::collider collider_type
  Element type.
- typedef ::xsd::cxx::tree::traits< collider_type, char > collider_traits
  Element traits type.
- const collider_type & collider () const
  Return a read-only (constant) reference to the element.
- collider_type & collider ()
  Return a read-write reference to the element.
- void collider (const collider_type &x)
  Set the element value.
- void collider (::std::unique_ptr< collider_type > p)
  Set the element value without copying.

events

Accessor and modifier functions for the events required element.

- typedef ::Common::events events_type
  Element type.
- typedef ::xsd::cxx::tree::traits< events_type, char > events_traits
  Element traits type.
- const events_type & events () const
  Return a read-only (constant) reference to the element.
- events_type & events ()
  Return a read-write reference to the element.
- void events (const events_type &x)
  Set the element value.
- void events (::std::unique_ptr< events_type > p)
  Set the element value without copying.

Constructors

- sentry (const className_type &, const resources_type &, const collider_type &, const events_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- sentry (const className_type &, ::std::unique_ptr< resources_type >, ::std::unique_ptr< collider_type >, ::std::unique_ptr< events_type >)
  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- sentry (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- sentry (const sentry &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual sentry * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- sentry & operator= (const sentry &x)
  Copy assignment operator.
- virtual ~sentry ()
  Destructor.

16.114.1 Detailed Description

Class corresponding to the sentry schema type.

16.114.2 Constructor & Destructor Documentation

16.114.2.1 sentry() [1/3]

```
Sentry::sentry::sentry (
            const className_type & className,
            ::std::unique_ptr< resources_type > resources,
            ::std::unique_ptr< collider_type > collider,
            ::std::unique_ptr< events_type > events )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.114.2.2 sentry() [2/3]

```
Sentry::sentry::sentry (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.114.2.3 sentry() [3/3]

```
Sentry::sentry::sentry (
            const sentry & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.114.3 Member Function Documentation

16.114.3.1 _clone()

```
sentry * Sentry::sentry::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.114.3.2 className() [1/4]

```
sentry::className_type & Sentry::sentry::className ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.114.3.3 className() [2/4]

```
const sentry::className_type & Sentry::sentry::className ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.114.3.4 className() [3/4]

```
void Sentry::sentry::className (
            ::std::unique_ptr< className_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.114.3.5 className() [4/4]

```
void Sentry::sentry::className (
            const className_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.114.3.6 collider() [1/4]

```
sentry::collider_type & Sentry::sentry::collider ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.114.3.7 collider() [2/4]

```
const sentry::collider_type & Sentry::sentry::collider ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.114.3.8 collider() [3/4]

```
void Sentry::sentry::collider (
            ::std::unique_ptr< collider_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.114.3.9 collider() [4/4]

```
void Sentry::sentry::collider (
            const collider_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.114.3.10 events() [1/4]

```
sentry::events_type & Sentry::sentry::events ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.114.3.11 events() [2/4]

```
const sentry::events_type & Sentry::sentry::events ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.114.3.12 events() [3/4]

```
void Sentry::sentry::events (
            ::std::unique_ptr< events_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.114.3.13 events() [4/4]

```
void Sentry::sentry::events (
            const events_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.114.3.14 operator=()

```
sentry & Sentry::sentry::operator= (
            const sentry & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.114.3.15 resources() [1/4]

```
sentry::resources_type & Sentry::sentry::resources ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.114.3.16 resources() [2/4]

```
const sentry::resources_type & Sentry::sentry::resources ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.114.3.17 resources() [3/4]

```
void Sentry::sentry::resources (
            ::std::unique_ptr< resources_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.114.3.18 resources() [4/4]

```
void Sentry::sentry::resources (
            const resources_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

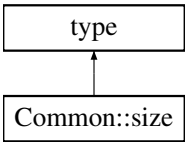
- Resources/XML/Generated/sentry.hxx
- Resources/XML/Generated/sentry.cxx

16.115 Common::size Class Reference

Class corresponding to the size schema type.

`#include <common.hxx>`

Inheritance diagram for Common::size:



width

Accessor and modifier functions for the width required element.

- typedef ::xml_schema::float_ width_type

  Element type.
- typedef ::xsd::cxx::tree::traits< width_type, char > width_traits

  Element traits type.
- const width_type & width () const

  Return a read-only (constant) reference to the element.
- width_type & width ()

  Return a read-write reference to the element.
- void width (const width_type &x)

  Set the element value.

height

Accessor and modifier functions for the height required element.

- typedef ::xml_schema::float_ height_type

  Element type.
- typedef ::xsd::cxx::tree::traits< height_type, char > height_traits

  Element traits type.
- const height_type & height () const

  Return a read-only (constant) reference to the element.
- height_type & height ()

  Return a read-write reference to the element.
- void height (const height_type &x)

  Set the element value.

Constructors

- size (const width_type &, const height_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- size (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- size (const size &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual size * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- size & operator= (const size &x)

  Copy assignment operator.
- virtual ~size ()

  Destructor.

16.115.1 Detailed Description

Class corresponding to the size schema type.

16.115.2 Constructor & Destructor Documentation

16.115.2.1 size() [1/2]

```
Common::size::size (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.115.2.2 size() [2/2]

```
Common::size::size (
            const size & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.115.3 Member Function Documentation

16.115.3.1 _clone()

```
size * Common::size::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.115.3.2 height() [1/3]

```
size::height_type & Common::size::height ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.115.3.3 height() [2/3]

```
const size::height_type & Common::size::height ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.115.3.4 height() [3/3]

```
void Common::size::height (
            const height_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.115.3.5 operator=()

```
size & Common::size::operator= (
            const size & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.115.3.6 width() [1/3]

```
size::width_type & Common::size::width ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.115.3.7 width() [2/3]

```
const size::width_type & Common::size::width ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.115.3.8 width() [3/3]

```
void Common::size::width (
            const width_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

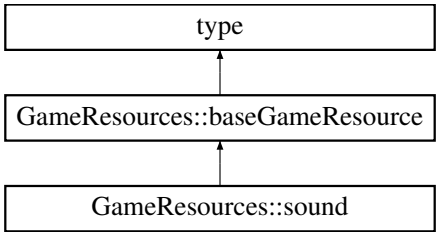
- Resources/XML/Generated/common.hxx
- Resources/XML/Generated/common.cxx

16.116 GameResources::sound Class Reference

Class corresponding to the sound schema type.

```
#include <resources.hxx>
```

Inheritance diagram for GameResources::sound:



Constructors

- sound (const name_type &, const path_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- sound (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- sound (const sound &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual sound * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- virtual ~sound ()

  Destructor.

Additional Inherited Members

16.116.1 Detailed Description

Class corresponding to the sound schema type.

16.116.2 Constructor & Destructor Documentation

16.116.2.1 sound() [1/2]

```
GameResources::sound::sound (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.116.2.2 sound() [2/2]

```
GameResources::sound::sound (
            const sound & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.116.3 Member Function Documentation

16.116.3.1 _clone()

```
sound * GameResources::sound::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

Reimplemented from GameResources::baseGameResource.

The documentation for this class was generated from the following files:

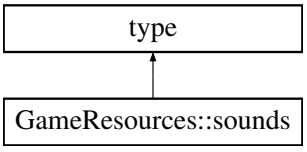
- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.117 GameResources::sounds Class Reference

Class corresponding to the sounds schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::sounds:



sound

Accessor and modifier functions for the sound sequence element.

- typedef ::GameResources::sound sound_type
  Element type.
- typedef ::xsd::cxx::tree::sequence< sound_type > sound_sequence
  Element sequence container type.
- typedef sound_sequence::iterator sound_iterator
  Element iterator type.
- typedef sound_sequence::const_iterator sound_const_iterator
  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< sound_type, char > sound_traits
  Element traits type.
- const sound_sequence & sound () const
  Return a read-only (constant) reference to the element sequence.
- sound_sequence & sound ()
  Return a read-write reference to the element sequence.
- void sound (const sound_sequence &s)
  Copy elements from a given sequence.

Constructors

- sounds ()
  Create an instance from the ultimate base and initializers for required elements and attributes.
- sounds (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- sounds (const sounds &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual sounds * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- sounds & operator= (const sounds &x)
  Copy assignment operator.
- virtual ~sounds ()
  Destructor.

16.117.1 Detailed Description

Class corresponding to the sounds schema type.

16.117.2 Constructor & Destructor Documentation

16.117.2.1 sounds() [1/2]

```
GameResources::sounds::sounds (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.117.2.2 sounds() [2/2]

```
GameResources::sounds::sounds (
            const sounds & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.117.3 Member Function Documentation

16.117.3.1 _clone()

```
sounds * GameResources::sounds::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.117.3.2 operator=()

```
sounds & GameResources::sounds::operator= (
            const sounds & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.117.3.3 sound() [1/3]

```
sounds::sound_sequence & GameResources::sounds::sound ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.117.3.4 sound() [2/3]

```
const sounds::sound_sequence & GameResources::sounds::sound ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.117.3.5 sound() [3/3]

```
void GameResources::sounds::sound (
            const sound_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

The documentation for this class was generated from the following files:

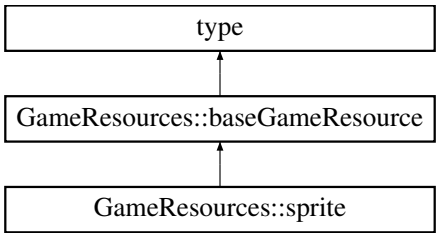
- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.118 GameResources::sprite Class Reference

Class corresponding to the sprite schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::sprite:



definition

Accessor and modifier functions for the definition optional element.

- typedef ::xml_schema::string definition_type

  Element type.
- typedef ::xsd::cxx::tree::optional< definition_type > definition_optional

  Element optional container type.
- typedef ::xsd::cxx::tree::traits< definition_type, char > definition_traits

  Element traits type.
- const definition_optional & definition () const

  Return a read-only (constant) reference to the element container.
- definition_optional & definition ()

  Return a read-write reference to the element container.
- void definition (const definition_type &x)

  Set the element value.
- void definition (const definition_optional &x)

  Set the element value.
- void definition (::std::unique_ptr< definition_type > p)

  Set the element value without copying.

size

Accessor and modifier functions for the size optional element.

- typedef ::Common::size size_type
  Element type.
- typedef ::xsd::cxx::tree::optional< size_type > size_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< size_type, char > size_traits
  Element traits type.
- const size_optional & size () const
  Return a read-only (constant) reference to the element container.
- size_optional & size ()
  Return a read-write reference to the element container.
- void size (const size_type &x)
  Set the element value.
- void size (const size_optional &x)
  Set the element value.
- void size (::std::unique_ptr< size_type > p)
  Set the element value without copying.

Constructors

- sprite (const name_type &, const path_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- sprite (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- sprite (const sprite &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual sprite * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- sprite & operator= (const sprite &x)
  Copy assignment operator.
- virtual ~sprite ()
  Destructor.

Additional Inherited Members

16.118.1 Detailed Description

Class corresponding to the sprite schema type.

16.118.2 Constructor & Destructor Documentation

16.118.2.1 sprite() [1/2]

```
GameResources::sprite::sprite (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.118.2.2 sprite() [2/2]

```
GameResources::sprite::sprite (
            const sprite & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.118.3 Member Function Documentation

16.118.3.1 _clone()

```
sprite * GameResources::sprite::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

Reimplemented from GameResources::baseGameResource.

16.118.3.2 definition() [1/5]

```
sprite::definition_optional & GameResources::sprite::definition ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.118.3.3 definition() [2/5]

```
const sprite::definition_optional & GameResources::sprite::definition ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.118.3.4 definition() [3/5]

```
void GameResources::sprite::definition (
            ::std::unique_ptr< definition_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.118.3.5 definition() [4/5]

```
void GameResources::sprite::definition (
            const definition_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.118.3.6 definition() [5/5]

```
void GameResources::sprite::definition (
            const definition_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.118.3.7 operator=()

```
sprite & GameResources::sprite::operator= (
            const sprite & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.118.3.8 size() [1/5]

```
sprite::size_optional & GameResources::sprite::size ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.118.3.9 size() [2/5]

```
const sprite::size_optional & GameResources::sprite::size ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.118.3.10 size() [3/5]

```
void GameResources::sprite::size (
            ::std::unique_ptr< size_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.118.3.11 size() [4/5]

```
void GameResources::sprite::size (
            const size_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.118.3.12 size() [5/5]

```
void GameResources::sprite::size (
            const size_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

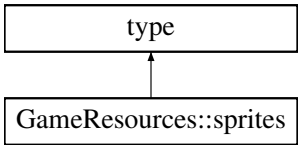
- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.119 GameResources::sprites Class Reference

Class corresponding to the sprites schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::sprites:



sprite

Accessor and modifier functions for the sprite sequence element.

- typedef ::GameResources::sprite sprite_type

  Element type.
- typedef ::xsd::cxx::tree::sequence< sprite_type > sprite_sequence

  Element sequence container type.
- typedef sprite_sequence::iterator sprite_iterator

  Element iterator type.
- typedef sprite_sequence::const_iterator sprite_const_iterator

  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< sprite_type, char > sprite_traits

  Element traits type.
- const sprite_sequence & sprite () const

  Return a read-only (constant) reference to the element sequence.
- sprite_sequence & sprite ()

  Return a read-write reference to the element sequence.
- void sprite (const sprite_sequence &s)

  Copy elements from a given sequence.

Constructors

- sprites ()

  Create an instance from the ultimate base and initializers for required elements and attributes.
- sprites (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- sprites (const sprites &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual sprites * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- sprites & operator= (const sprites &x)

  Copy assignment operator.
- virtual ~sprites ()

  Destructor.

16.119.1 Detailed Description

Class corresponding to the sprites schema type.

16.119.2 Constructor & Destructor Documentation

16.119.2.1 sprites() [1/2]

```
GameResources::sprites::sprites (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.119.2.2 sprites() [2/2]

```
GameResources::sprites::sprites (
            const sprites & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.119.3 Member Function Documentation

16.119.3.1 _clone()

```
sprites * GameResources::sprites::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.119.3.2 operator=()

```
sprites & GameResources::sprites::operator= (
            const sprites & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.119.3.3 sprite() [1/3]

```
sprites::sprite_sequence & GameResources::sprites::sprite ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.119.3.4 sprite() [2/3]

```
const sprites::sprite_sequence & GameResources::sprites::sprite ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.119.3.5 sprite() [3/3]

```
void GameResources::sprites::sprite (
            const sprite_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.120 Spritesheet Class Reference

Public Member Functions

- **Spritesheet** (const std::string &path, std::string &spriteSheetid, int width, int height, SDL_Renderer ∗renderer)

- **Spritesheet** (const std::string &path, const std::string &jsonPath, std::string &spriteSheetId, SDL_Renderer ∗renderer)
- void **draw_selected_sprite** (float x, float y, float scale=1, float rotation=0)
- void **select_sprite** (int x, int y)
- void **select_sprite** (const std::string &spriteName)

The documentation for this class was generated from the following files:

- Engine/Rendering/Spritesheet.hpp
- Engine/Rendering/Spritesheet.cpp

16.121 System< C > Class Template Reference

Public Attributes

- std::multimap< EntityId, C ∗ > **components**

The documentation for this class was generated from the following file:

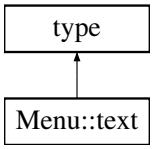
- Game/Components/Component.hpp

16.122 Menu::text Class Reference

Class corresponding to the text schema type.

`#include <menu.hxx>`

Inheritance diagram for Menu::text:



position

Accessor and modifier functions for the position required element.

- typedef ::Common::position position_type
  Element type.
- typedef ::xsd::cxx::tree::traits< position_type, char > position_traits
  Element traits type.
- const position_type & position () const
  Return a read-only (constant) reference to the element.
- position_type & position ()
  Return a read-write reference to the element.
- void position (const position_type &x)
  Set the element value.
- void position (::std::unique_ptr< position_type > p)
  Set the element value without copying.

color

Accessor and modifier functions for the color required element.

- typedef ::Common::color color_type
  Element type.
- typedef ::xsd::cxx::tree::traits< color_type, char > color_traits
  Element traits type.
- const color_type & color () const
  Return a read-only (constant) reference to the element.
- color_type & color ()
  Return a read-write reference to the element.
- void color (const color_type &x)
  Set the element value.
- void color (::std::unique_ptr< color_type > p)
  Set the element value without copying.

font

Accessor and modifier functions for the font required element.

- typedef ::Common::font font_type
  Element type.
- typedef ::xsd::cxx::tree::traits< font_type, char > font_traits
  Element traits type.
- const font_type & font () const
  Return a read-only (constant) reference to the element.
- font_type & font ()
  Return a read-write reference to the element.
- void font (const font_type &x)
  Set the element value.
- void font (::std::unique_ptr< font_type > p)
  Set the element value without copying.

content

Accessor and modifier functions for the content required element.

- typedef ::xml_schema::string content_type
  Element type.
- typedef ::xsd::cxx::tree::traits< content_type, char > content_traits
  Element traits type.
- const content_type & content () const
  Return a read-only (constant) reference to the element.
- content_type & content ()
  Return a read-write reference to the element.
- void content (const content_type &x)
  Set the element value.
- void content (::std::unique_ptr< content_type > p)
  Set the element value without copying.

Constructors

- text (const position_type &, const color_type &, const font_type &, const content_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- text (::std::unique_ptr< position_type >, ::std::unique_ptr< color_type >, ::std::unique_ptr< font_type >, const content_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- text (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- text (const text &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual text * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- text & operator= (const text &x)

  Copy assignment operator.
- virtual ~text ()

  Destructor.

16.122.1 Detailed Description

Class corresponding to the text schema type.

16.122.2 Constructor & Destructor Documentation

16.122.2.1 text() [1/3]

```
Menu::text::text (
            ::std::unique_ptr< position_type > position,
            ::std::unique_ptr< color_type > color,
            ::std::unique_ptr< font_type > font,
            const content_type & content )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.122.2.2 text() [2/3]

```
Menu::text::text (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.122.2.3 text() [3/3]

```
Menu::text::text (
            const text & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.122.3 Member Function Documentation

16.122.3.1 _clone()

```
text * Menu::text::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.122.3.2 color() [1/4]

```
text::color_type & Menu::text::color ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.122.3.3 color() [2/4]

```
const text::color_type & Menu::text::color ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.122.3.4 color() [3/4]

```
void Menu::text::color (
            ::std::unique_ptr< color_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.122.3.5 color() [4/4]

```
void Menu::text::color (
            const color_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.122.3.6 content() [1/4]

```
text::content_type & Menu::text::content ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.122.3.7 content() [2/4]

```
const text::content_type & Menu::text::content ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.122.3.8 content() [3/4]

```
void Menu::text::content (
            ::std::unique_ptr< content_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.122.3.9 content() [4/4]

```
void Menu::text::content (
            const content_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.122.3.10 font() [1/4]

```
text::font_type & Menu::text::font ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.122.3.11 font() [2/4]

```
const text::font_type & Menu::text::font ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.122.3.12 font() [3/4]

```
void Menu::text::font (
            ::std::unique_ptr< font_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.122.3.13 font() [4/4]

```
void Menu::text::font (
            const font_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.122.3.14 operator=()

```
text & Menu::text::operator= (
            const text & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.122.3.15 position() [1/4]

```
text::position_type & Menu::text::position ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.122.3.16 position() [2/4]

```
const text::position_type & Menu::text::position ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.122.3.17 position() [3/4]

```
void Menu::text::position (
            ::std::unique_ptr< position_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.122.3.18 position() [4/4]

```
void Menu::text::position (
            const position_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

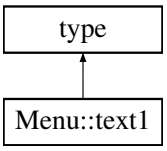
- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.123 Menu::text1 Class Reference

Class corresponding to the text1 schema type.

```
#include <menu.hxx>
```

Inheritance diagram for Menu::text1:



font

Accessor and modifier functions for the font required element.

- typedef ::Common::font font_type

  Element type.
- typedef ::xsd::cxx::tree::traits< font_type, char > font_traits

  Element traits type.
- const font_type & font () const

Return a read-only (constant) reference to the element.

- font_type & font ()

  Return a read-write reference to the element.

- void font (const font_type &x)

  Set the element value.

- void font (::std::unique_ptr< font_type > p)

  Set the element value without copying.

color

Accessor and modifier functions for the color required element.

- typedef ::Common::color color_type

  Element type.

- typedef ::xsd::cxx::tree::traits< color_type, char > color_traits

  Element traits type.

- const color_type & color () const

  Return a read-only (constant) reference to the element.

- color_type & color ()

  Return a read-write reference to the element.

- void color (const color_type &x)

  Set the element value.

- void color (::std::unique_ptr< color_type > p)

  Set the element value without copying.

content

Accessor and modifier functions for the content required element.

- typedef ::xml_schema::string content_type

  Element type.

- typedef ::xsd::cxx::tree::traits< content_type, char > content_traits

  Element traits type.

- const content_type & content () const

  Return a read-only (constant) reference to the element.

- content_type & content ()

  Return a read-write reference to the element.

- void content (const content_type &x)

  Set the element value.

- void content (::std::unique_ptr< content_type > p)

  Set the element value without copying.

Constructors

- text1 (const font_type &, const color_type &, const content_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- text1 (::std::unique_ptr< font_type >, ::std::unique_ptr< color_type >, const content_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- text1 (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- text1 (const text1 &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual text1 * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- text1 & operator= (const text1 &x)

  Copy assignment operator.
- virtual ~text1 ()

  Destructor.

16.123.1 Detailed Description

Class corresponding to the text1 schema type.

16.123.2 Constructor & Destructor Documentation

16.123.2.1 text1() [1/3]

```
Menu::text1::text1 (
            ::std::unique_ptr< font_type > font,
            ::std::unique_ptr< color_type > color,
            const content_type & content )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.123.2.2 text1() [2/3]

```
Menu::text1::text1 (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.123.2.3 text1() [3/3]

```
Menu::text1::text1 (
            const text1 & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.123.3 Member Function Documentation

16.123.3.1 _clone()

```
text1 * Menu::text1::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.123.3.2 color() [1/4]

```
text1::color_type & Menu::text1::color ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.123.3.3 color() [2/4]

```
const text1::color_type & Menu::text1::color ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.123.3.4 color() [3/4]

```
void Menu::text1::color (
            ::std::unique_ptr< color_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.123.3.5 color() [4/4]

```
void Menu::text1::color (
            const color_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.123.3.6 content() [1/4]

```
text1::content_type & Menu::text1::content ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.123.3.7 content() [2/4]

```
const text1::content_type & Menu::text1::content ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.123.3.8 content() [3/4]

```
void Menu::text1::content (
            ::std::unique_ptr< content_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.123.3.9 content() [4/4]

```
void Menu::text1::content (
            const content_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.123.3.10 font() [1/4]

```
text1::font_type & Menu::text1::font ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.123.3.11 font() [2/4]

```
const text1::font_type & Menu::text1::font ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.123.3.12 font() [3/4]

```
void Menu::text1::font (
            ::std::unique_ptr< font_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.123.3.13 font() [4/4]

```
void Menu::text1::font (
            const font_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.123.3.14 operator=()

```
text1 & Menu::text1::operator= (
             const text1 & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

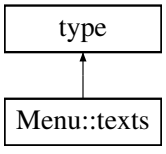
- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.124 Menu::texts Class Reference

Class corresponding to the texts schema type.

```
#include <menu.hxx>
```

Inheritance diagram for Menu::texts:



text

Accessor and modifier functions for the text sequence element.

- typedef ::Menu::text text_type
  Element type.
- typedef ::xsd::cxx::tree::sequence< text_type > text_sequence
  Element sequence container type.
- typedef text_sequence::iterator text_iterator
  Element iterator type.
- typedef text_sequence::const_iterator text_const_iterator
  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< text_type, char > text_traits
  Element traits type.
- const text_sequence & text () const
  Return a read-only (constant) reference to the element sequence.
- text_sequence & text ()
  Return a read-write reference to the element sequence.
- void text (const text_sequence &s)
  Copy elements from a given sequence.

Constructors

- texts ()
  Create an instance from the ultimate base and initializers for required elements and attributes.
- texts (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- texts (const texts &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual texts * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- texts & operator= (const texts &x)
  Copy assignment operator.
- virtual ~texts ()
  Destructor.

16.124.1 Detailed Description

Class corresponding to the texts schema type.

16.124.2 Constructor & Destructor Documentation

16.124.2.1 texts() [1/2]

```
Menu::texts::texts (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.124.2.2 texts() [2/2]

```
Menu::texts::texts (
            const texts & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.124.3 Member Function Documentation

16.124.3.1 _clone()

```
texts * Menu::texts::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.124.3.2 operator=()

```
texts & Menu::texts::operator= (
            const texts & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.124.3.3 text() [1/3]

```
texts::text_sequence & Menu::texts::text ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.124.3.4 text() [2/3]

```
const texts::text_sequence & Menu::texts::text ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.124.3.5 text() [3/3]

```
void Menu::texts::text (
            const text_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

The documentation for this class was generated from the following files:

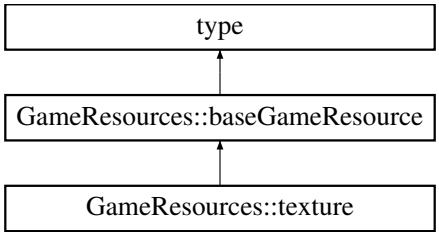
- Resources/XML/Generated/menu.hxx
- Resources/XML/Generated/menu.cxx

16.125 GameResources::texture Class Reference

Class corresponding to the texture schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::texture:



Constructors

- texture (const name_type &, const path_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- texture (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- texture (const texture &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual texture * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- virtual ~texture ()

  Destructor.

Additional Inherited Members

16.125.1 Detailed Description

Class corresponding to the texture schema type.

16.125.2 Constructor & Destructor Documentation

16.125.2.1 texture() [1/2]

```
GameResources::texture::texture (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.125.2.2 texture() [2/2]

```
GameResources::texture::texture (
            const texture & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.125.3 Member Function Documentation

16.125.3.1 _clone()

```
texture * GameResources::texture::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

Reimplemented from GameResources::baseGameResource.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.126 TextureManager Class Reference

Public Member Functions

- SDL_Texture * **GetTexture** (std::string textureId)
- bool **CreateTexture** (SDL_Surface *surface, const std::string &textureId, SDL_Renderer *renderer)
- bool **load** (const std::string &path, const std::string &textureId)
- void **draw** (std::string &textureId, int x, int y, int w, int h, double scale, double r, SDL_Renderer *renderer, SDL_RendererFlip=SDL_FLIP_NONE)
- void **drawFrame** (std::string &id, float x, float y, int width, int height, int currentRow, int currentFrame, SDL_Renderer *pRenderer, SDL_RendererFlip flip, float rotation)
- void **drawFrame** (std::string &id, SDL_Rect *srcRect, float x, float y, SDL_Renderer *pRenderer, SDL_RendererFlip flip, float scale=1, float rotation=0, SDL_FPoint *pivot=nullptr)
- void **clearFromTextureMap** (std::string &id)
- Vector2 **getDimensions** (const std::string &id)

Static Public Member Functions

- static TextureManager * **GetInstance** ()

The documentation for this class was generated from the following files:

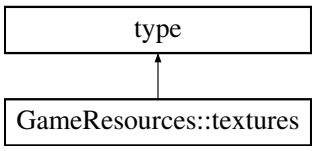
- Engine/Rendering/TextureManager.hpp
- Engine/Rendering/TextureManager.cpp

16.127 GameResources::textures Class Reference

Class corresponding to the textures schema type.

`#include <resources.hxx>`

Inheritance diagram for GameResources::textures:



texture

Accessor and modifier functions for the texture sequence element.

- typedef ::GameResources::texture texture_type
  Element type.
- typedef ::xsd::cxx::tree::sequence< texture_type > texture_sequence
  Element sequence container type.
- typedef texture_sequence::iterator texture_iterator
  Element iterator type.
- typedef texture_sequence::const_iterator texture_const_iterator
  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< texture_type, char > texture_traits
  Element traits type.
- const texture_sequence & texture () const
  Return a read-only (constant) reference to the element sequence.
- texture_sequence & texture ()
  Return a read-write reference to the element sequence.
- void texture (const texture_sequence &s)
  Copy elements from a given sequence.

Constructors

- textures ()
  Create an instance from the ultimate base and initializers for required elements and attributes.
- textures (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- textures (const textures &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual textures * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- textures & operator= (const textures &x)
  Copy assignment operator.
- virtual ~textures ()
  Destructor.

16.127.1 Detailed Description

Class corresponding to the textures schema type.

16.127.2 Constructor & Destructor Documentation

16.127.2.1 textures() [1/2]

```
GameResources::textures::textures (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.127.2.2 textures() [2/2]

```
GameResources::textures::textures (
            const textures & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.127.3 Member Function Documentation

16.127.3.1 _clone()

```
textures * GameResources::textures::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.127.3.2 operator=()

```
textures & GameResources::textures::operator= (
            const textures & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.127.3.3 texture() [1/3]

```
textures::texture_sequence & GameResources::textures::texture ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.127.3.4 texture() [2/3]

```
const textures::texture_sequence & GameResources::textures::texture ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.127.3.5 texture() [3/3]

```
void GameResources::textures::texture (
            const texture_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/resources.hxx
- Resources/XML/Generated/resources.cxx

16.128 TextWrapper Class Reference

Public Member Functions

- Vector2 **getSize** ()
- std::string **getTextureId** ()
- void **render** (float x, float y, int width, int height, double scale, double r)
- void **render** (float x, float y)

Static Public Member Functions

- static TextWrapper * **createText** (const RenderingAPI &renderingAPI, const std::string &text, const std::string &fontPath, int fontSize, const std::string &hex, const std::string &textureId)

The documentation for this class was generated from the following files:

- Engine/Rendering/TextWrapper.hpp
- Engine/Rendering/TextWrapper.cpp

16.129 TMXLevel Class Reference

Public Member Functions

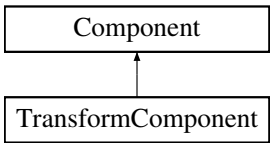
- **TMXLevel** (const char *tmxPath, const char *spritesheetPath, const char *spritesheetId, const RenderingAPI &renderingApi, PhysicsEngineAdapter &physicsEngineAdapter)
- void **render** (RenderingAPI &renderingApi)
- void **cleanup** ()
- void **getObjectPositions** (const std::multimap< std::string, Components::component * > &outEntities)
- void **initStaticCollision** ()

The documentation for this class was generated from the following files:

- Engine/Rendering/TMXLevel.hpp
- Engine/Rendering/TMXLevel.cpp

16.130 TransformComponent Class Reference

Inheritance diagram for TransformComponent:



Public Member Functions

- void **render** () override
- void **update** (const Input &inputSystem) override
- void **refLocation** (const float &rX, const float &rY)
- void **setRotation** (float r)
- **TransformComponent** (EntityId id)
- void **fixedUpdate** (const float &deltaTime) override
- Component * **build** (EntityId entityId, const Components::component *component) override
- void **initialize** (EntityObject &entityParent) override
- Vector2 **right** () const
- Vector2 **left** () const
- Vector2 **up** () const
- Vector2 **down** () const
- Vector2 **getPosition** () const
- std::string **name** () const override
- TransformComponent & **operator=** (Vector2 &v2)

Public Attributes

- float **rotation** = 0

Additional Inherited Members

The documentation for this class was generated from the following files:

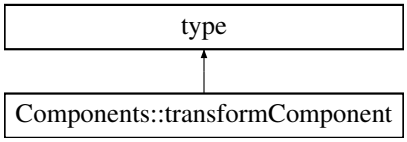
- Game/Components/TransformComponent.hpp
- Game/Components/TransformComponent.cpp

16.131 Components::transformComponent Class Reference

Class corresponding to the transformComponent schema type.

```
#include <components.hxx>
```

Inheritance diagram for Components::transformComponent:



position

Accessor and modifier functions for the position required element.

- typedef ::Common::position position_type

  Element type.
- typedef ::xsd::cxx::tree::traits< position_type, char > position_traits

  Element traits type.
- const position_type & position () const

  Return a read-only (constant) reference to the element.
- position_type & position ()

  Return a read-write reference to the element.
- void position (const position_type &x)

  Set the element value.
- void position (::std::unique_ptr< position_type > p)

  Set the element value without copying.

Constructors

- transformComponent (const position_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- transformComponent (::std::unique_ptr< position_type >)

  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- transformComponent (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- transformComponent (const transformComponent &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual transformComponent * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- transformComponent & operator= (const transformComponent &x)

  Copy assignment operator.
- virtual ~transformComponent ()

  Destructor.

16.131.1 Detailed Description

Class corresponding to the transformComponent schema type.

16.131.2 Constructor & Destructor Documentation

16.131.2.1 transformComponent() [1/3]

```
Components::transformComponent::transformComponent (
            ::std::unique_ptr< position_type > position )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.131.2.2 transformComponent() [2/3]

```
Components::transformComponent::transformComponent (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.131.2.3 transformComponent() [3/3]

```
Components::transformComponent::transformComponent (
            const transformComponent & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.131.3 Member Function Documentation

16.131.3.1 _clone()

```
transformComponent * Components::transformComponent::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.131.3.2 operator=()

```
transformComponent & Components::transformComponent::operator= (
            const transformComponent & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.131.3.3 position() [1/4]

```
transformComponent::position_type & Components::transformComponent::position ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.131.3.4 position() [2/4]

```
const transformComponent::position_type & Components::transformComponent::position ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.131.3.5 position() [3/4]

```
void Components::transformComponent::position (
            ::std::unique_ptr< position_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.131.3.6 position() [4/4]

```
void Components::transformComponent::position (
            const position_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

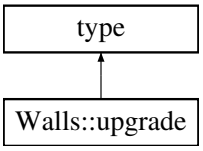
- Resources/XML/Generated/components.hxx
- Resources/XML/Generated/components.cxx

16.132 Walls::upgrade Class Reference

Class corresponding to the upgrade schema type.

`#include <wall.hxx>`

Inheritance diagram for Walls::upgrade:



cost

Accessor and modifier functions for the cost required element.

- typedef ::xml_schema::int_ cost_type
  Element type.
- typedef ::xsd::cxx::tree::traits< cost_type, char > cost_traits
  Element traits type.
- const cost_type & cost () const
  Return a read-only (constant) reference to the element.
- cost_type & cost ()
  Return a read-write reference to the element.
- void cost (const cost_type &x)
  Set the element value.

nextLevel

Accessor and modifier functions for the nextLevel required element.

- typedef ::xml_schema::idref nextLevel_type
  Element type.
- typedef ::xsd::cxx::tree::traits< nextLevel_type, char > nextLevel_traits
  Element traits type.
- const nextLevel_type & nextLevel () const
  Return a read-only (constant) reference to the element.
- nextLevel_type & nextLevel ()
  Return a read-write reference to the element.
- void nextLevel (const nextLevel_type &x)
  Set the element value.
- void nextLevel (::std::unique_ptr< nextLevel_type > p)
  Set the element value without copying.

Constructors

- upgrade (const cost_type &, const nextLevel_type &)
  Create an instance from the ultimate base and initializers for required elements and attributes.
- upgrade (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Create an instance from a DOM element.
- upgrade (const upgrade &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)
  Copy constructor.
- virtual upgrade * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const
  Copy the instance polymorphically.
- upgrade & operator= (const upgrade &x)
  Copy assignment operator.
- virtual ~upgrade ()
  Destructor.

16.132.1 Detailed Description

Class corresponding to the upgrade schema type.

16.132.2 Constructor & Destructor Documentation

16.132.2.1 upgrade() [1/2]

```
Walls::upgrade::upgrade (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.132.2.2 upgrade() [2/2]

```
Walls::upgrade::upgrade (
            const upgrade & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.132.3 Member Function Documentation

16.132.3.1 _clone()

```
upgrade * Walls::upgrade::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.132.3.2 cost() [1/3]

```
upgrade::cost_type & Walls::upgrade::cost ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.132.3.3 cost() [2/3]

```
const upgrade::cost_type & Walls::upgrade::cost ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.132.3.4 cost() [3/3]

```
void Walls::upgrade::cost (
            const cost_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.132.3.5 nextLevel() [1/4]

```
upgrade::nextLevel_type & Walls::upgrade::nextLevel ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.132.3.6 nextLevel() [2/4]

```
const upgrade::nextLevel_type & Walls::upgrade::nextLevel ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.132.3.7 nextLevel() [3/4]

```
void Walls::upgrade::nextLevel (
            ::std::unique_ptr< nextLevel_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.132.3.8 nextLevel() [4/4]

```
void Walls::upgrade::nextLevel (
            const nextLevel_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.132.3.9 operator=()

```
upgrade & Walls::upgrade::operator= (
            const upgrade & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/wall.hxx
- Resources/XML/Generated/wall.cxx

16.133 Vector2 Struct Reference

Public Member Functions

- **Vector2** (float x, float y)
- Vector2 **operator+** (const Vector2 &other) const
- Vector2 **operator-** (const Vector2 &other) const
- Vector2 **operator∗** (float scale) const
- Vector2 **operator∗** (const Vector2 &other)
- bool **operator==** (const Vector2 &other)

Public Attributes

- float **x**
- float **y**

The documentation for this struct was generated from the following file:

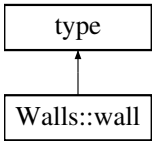
- API/Helpers/Vector2.hpp

16.134 Walls::wall Class Reference

Class corresponding to the wall schema type.

`#include <wall.hxx>`

Inheritance diagram for Walls::wall:



name

Accessor and modifier functions for the name required element.

- typedef ::xml_schema::string name_type
  Element type.
- typedef ::xsd::cxx::tree::traits< name_type, char > name_traits
  Element traits type.
- const name_type & name () const
  Return a read-only (constant) reference to the element.
- name_type & name ()
  Return a read-write reference to the element.
- void name (const name_type &x)
  Set the element value.
- void name (::std::unique_ptr< name_type > p)
  Set the element value without copying.

level

Accessor and modifier functions for the level required element.

- typedef ::xml_schema::id level_type
  Element type.
- typedef ::xsd::cxx::tree::traits< level_type, char > level_traits
  Element traits type.
- const level_type & level () const
  Return a read-only (constant) reference to the element.
- level_type & level ()
  Return a read-write reference to the element.
- void level (const level_type &x)
  Set the element value.
- void level (::std::unique_ptr< level_type > p)
  Set the element value without copying.

baseHealth

Accessor and modifier functions for the baseHealth required element.

- typedef ::xml_schema::int_ baseHealth_type
  Element type.
- typedef ::xsd::cxx::tree::traits< baseHealth_type, char > baseHealth_traits
  Element traits type.
- const baseHealth_type & baseHealth () const
  Return a read-only (constant) reference to the element.
- baseHealth_type & baseHealth ()
  Return a read-write reference to the element.
- void baseHealth (const baseHealth_type &x)
  Set the element value.

pricing

Accessor and modifier functions for the pricing required element.

- typedef ::Walls::pricing pricing_type
  Element type.
- typedef ::xsd::cxx::tree::traits< pricing_type, char > pricing_traits
  Element traits type.
- const pricing_type & pricing () const
  Return a read-only (constant) reference to the element.
- pricing_type & pricing ()
  Return a read-write reference to the element.
- void pricing (const pricing_type &x)
  Set the element value.
- void pricing (::std::unique_ptr< pricing_type > p)
  Set the element value without copying.

powers

Accessor and modifier functions for the powers optional element.

- typedef ::Walls::powers powers_type
  Element type.
- typedef ::xsd::cxx::tree::optional< powers_type > powers_optional
  Element optional container type.
- typedef ::xsd::cxx::tree::traits< powers_type, char > powers_traits
  Element traits type.
- const powers_optional & powers () const
  Return a read-only (constant) reference to the element container.
- powers_optional & powers ()
  Return a read-write reference to the element container.
- void powers (const powers_type &x)
  Set the element value.
- void powers (const powers_optional &x)
  Set the element value.
- void powers (::std::unique_ptr< powers_type > p)
  Set the element value without copying.

resources

Accessor and modifier functions for the resources required element.

- typedef ::Common::resources resources_type
  Element type.
- typedef ::xsd::cxx::tree::traits< resources_type, char > resources_traits
  Element traits type.
- const resources_type & resources () const
  Return a read-only (constant) reference to the element.
- resources_type & resources ()
  Return a read-write reference to the element.
- void resources (const resources_type &x)
  Set the element value.
- void resources (::std::unique_ptr< resources_type > p)
  Set the element value without copying.

events

Accessor and modifier functions for the events required element.

- typedef ::Common::events events_type

  Element type.
- typedef ::xsd::cxx::tree::traits< events_type, char > events_traits

  Element traits type.
- const events_type & events () const

  Return a read-only (constant) reference to the element.
- events_type & events ()

  Return a read-write reference to the element.
- void events (const events_type &x)

  Set the element value.
- void events (::std::unique_ptr< events_type > p)

  Set the element value without copying.

Constructors

- wall (const name_type &, const level_type &, const baseHealth_type &, const pricing_type &, const resources_type &, const events_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- wall (const name_type &, const level_type &, const baseHealth_type &, ::std::unique_ptr< pricing_type >, ::std::unique_ptr< resources_type >, ::std::unique_ptr< events_type >)

  Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).
- wall (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- wall (const wall &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual wall * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- wall & operator= (const wall &x)

  Copy assignment operator.
- virtual ~wall ()

  Destructor.

16.134.1 Detailed Description

Class corresponding to the wall schema type.

16.134.2 Constructor & Destructor Documentation

16.134.2.1 wall() [1/3]

```
Walls::wall::wall (
            const name_type & name,
            const level_type & level,
            const baseHealth_type & baseHealth,
            ::std::unique_ptr< pricing_type > pricing,
            ::std::unique_ptr< resources_type > resources,
            ::std::unique_ptr< events_type > events )
```

Create an instance from the ultimate base and initializers for required elements and attributes (::std::unique_ptr version).

This constructor will try to use the passed values directly instead of making copies.

16.134.2.2 wall() [2/3]

```
Walls::wall::wall (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.134.2.3 wall() [3/3]

```
Walls::wall::wall (
            const wall & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.134.3 Member Function Documentation

16.134.3.1 _clone()

```
wall * Walls::wall::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.134.3.2 baseHealth() [1/3]

```
wall::baseHealth_type & Walls::wall::baseHealth ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.134.3.3 baseHealth() [2/3]

```
const wall::baseHealth_type & Walls::wall::baseHealth ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.134.3.4 baseHealth() [3/3]

```
void Walls::wall::baseHealth (
            const baseHealth_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.134.3.5 events() [1/4]

```
wall::events_type & Walls::wall::events ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.134.3.6 events() [2/4]

```
const wall::events_type & Walls::wall::events ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.134.3.7 events() [3/4]

```
void Walls::wall::events (
            ::std::unique_ptr< events_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.134.3.8 events() [4/4]

```
void Walls::wall::events (
            const events_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.134.3.9 level() [1/4]

```
wall::level_type & Walls::wall::level ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.134.3.10 level() [2/4]

```
const wall::level_type & Walls::wall::level ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.134.3.11 level() [3/4]

```
void Walls::wall::level (
            ::std::unique_ptr< level_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.134.3.12 level() [4/4]

```
void Walls::wall::level (
```

```
            const level_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.134.3.13 name() [1/4]

```
wall::name_type & Walls::wall::name ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.134.3.14 name() [2/4]

```
const wall::name_type & Walls::wall::name ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.134.3.15 name() [3/4]

```
void Walls::wall::name (
            ::std::unique_ptr< name_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.134.3.16 name() [4/4]

```
void Walls::wall::name (
            const name_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.134.3.17 operator=()

```
wall & Walls::wall::operator= (
            const wall & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.134.3.18 powers() [1/5]

```
wall::powers_optional & Walls::wall::powers ( )
```

Return a read-write reference to the element container.

Returns

A reference to the optional container.

16.134.3.19 powers() [2/5]

```
const wall::powers_optional & Walls::wall::powers ( ) const
```

Return a read-only (constant) reference to the element container.

Returns

A constant reference to the optional container.

16.134.3.20 powers() [3/5]

```
void Walls::wall::powers (
            ::std::unique_ptr< powers_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.134.3.21 powers() [4/5]

```
void Walls::wall::powers (
            const powers_optional & x )
```

Set the element value.

Parameters

x	An optional container with the new value to set.

If the value is present in *x* then this function makes a copy of this value and sets it as the new value of the element. Otherwise the element container is set the 'not present' state.

16.134.3.22 powers() [5/5]

```
void Walls::wall::powers (
            const powers_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.134.3.23 pricing() [1/4]

```
wall::pricing_type & Walls::wall::pricing ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.134.3.24 pricing() [2/4]

```
const wall::pricing_type & Walls::wall::pricing ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.134.3.25 pricing() [3/4]

```
void Walls::wall::pricing (
            ::std::unique_ptr< pricing_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.134.3.26 pricing() [4/4]

```
void Walls::wall::pricing (
            const pricing_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.134.3.27 resources() [1/4]

```
wall::resources_type & Walls::wall::resources ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.134.3.28 resources() [2/4]

```
const wall::resources_type & Walls::wall::resources ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.134.3.29 resources() [3/4]

```
void Walls::wall::resources (
            ::std::unique_ptr< resources_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.134.3.30 resources() [4/4]

```
void Walls::wall::resources (
            const resources_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

The documentation for this class was generated from the following files:

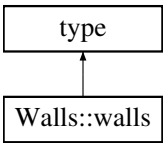
- Resources/XML/Generated/wall.hxx
- Resources/XML/Generated/wall.cxx

16.135 Walls::walls Class Reference

Class corresponding to the walls schema type.

```
#include <wall.hxx>
```

Inheritance diagram for Walls::walls:



className

Accessor and modifier functions for the className required element.

- typedef ::xml_schema::string className_type

  Element type.
- typedef ::xsd::cxx::tree::traits< className_type, char > className_traits

  Element traits type.
- const className_type & className () const

  Return a read-only (constant) reference to the element.
- className_type & className ()

  Return a read-write reference to the element.
- void className (const className_type &x)

  Set the element value.
- void className (::std::unique_ptr< className_type > p)

  Set the element value without copying.

wall

Accessor and modifier functions for the wall sequence element.

- typedef ::Walls::wall wall_type

  Element type.
- typedef ::xsd::cxx::tree::sequence< wall_type > wall_sequence

  Element sequence container type.
- typedef wall_sequence::iterator wall_iterator

  Element iterator type.
- typedef wall_sequence::const_iterator wall_const_iterator

  Element constant iterator type.
- typedef ::xsd::cxx::tree::traits< wall_type, char > wall_traits

  Element traits type.
- const wall_sequence & wall () const

  Return a read-only (constant) reference to the element sequence.
- wall_sequence & wall ()

  Return a read-write reference to the element sequence.
- void wall (const wall_sequence &s)

  Copy elements from a given sequence.

Constructors

- walls (const className_type &)

  Create an instance from the ultimate base and initializers for required elements and attributes.
- walls (const ::xercesc::DOMElement &e, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Create an instance from a DOM element.
- walls (const walls &x, ::xml_schema::flags f=0, ::xml_schema::container *c=0)

  Copy constructor.
- virtual walls * _clone (::xml_schema::flags f=0, ::xml_schema::container *c=0) const

  Copy the instance polymorphically.
- walls & operator= (const walls &x)

  Copy assignment operator.
- virtual ~walls ()

  Destructor.

16.135.1 Detailed Description

Class corresponding to the walls schema type.

16.135.2 Constructor & Destructor Documentation

16.135.2.1 walls() [1/2]

```
Walls::walls::walls (
            const ::xercesc::DOMElement & e,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Create an instance from a DOM element.

Parameters

e	A DOM element to extract the data from.
f	Flags to create the new instance with.
c	A pointer to the object that will contain the new instance.

16.135.2.2 walls() [2/2]

```
Walls::walls::walls (
            const walls & x,
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 )
```

Copy constructor.

Parameters

x	An instance to make a copy of.
f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

For polymorphic object models use the `_clone` function instead.

16.135.3 Member Function Documentation

16.135.3.1 _clone()

```
walls * Walls::walls::_clone (
            ::xml_schema::flags f = 0,
            ::xml_schema::container * c = 0 ) const  [virtual]
```

Copy the instance polymorphically.

Parameters

f	Flags to create the copy with.
c	A pointer to the object that will contain the copy.

Returns

A pointer to the dynamically allocated copy.

This function ensures that the dynamic type of the instance is used for copying and should be used for polymorphic object models instead of the copy constructor.

16.135.3.2 className() [1/4]

```
walls::className_type & Walls::walls::className ( )
```

Return a read-write reference to the element.

Returns

A reference to the element.

16.135.3.3 className() [2/4]

```
const walls::className_type & Walls::walls::className ( ) const
```

Return a read-only (constant) reference to the element.

Returns

A constant reference to the element.

16.135.3.4 className() [3/4]

```
void Walls::walls::className (
            ::std::unique_ptr< className_type > p )
```

Set the element value without copying.

Parameters

p	A new value to use.

This function will try to use the passed value directly instead of making a copy.

16.135.3.5 className() [4/4]

```
void Walls::walls::className (
            const className_type & x )
```

Set the element value.

Parameters

x	A new value to set.

This function makes a copy of its argument and sets it as the new value of the element.

16.135.3.6 operator=()

```
walls & Walls::walls::operator= (
            const walls & x )
```

Copy assignment operator.

Parameters

x	An instance to make a copy of.

Returns

A reference to itself.

For polymorphic object models use the `_clone` function instead.

16.135.3.7 wall() [1/3]

```
walls::wall_sequence & Walls::walls::wall ( )
```

Return a read-write reference to the element sequence.

Returns

A reference to the sequence container.

16.135.3.8 wall() [2/3]

```
const walls::wall_sequence & Walls::walls::wall ( ) const
```

Return a read-only (constant) reference to the element sequence.

Returns

A constant reference to the sequence container.

16.135.3.9 wall() [3/3]

```
void Walls::walls::wall (
            const wall_sequence & s )
```

Copy elements from a given sequence.

Parameters

s	A sequence to copy elements from.

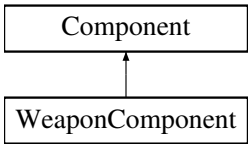
For each element in *s* this function makes a copy and adds it to the sequence. Note that this operation completely changes the sequence and all old elements will be lost.

The documentation for this class was generated from the following files:

- Resources/XML/Generated/wall.hxx
- Resources/XML/Generated/wall.cxx

16.136 WeaponComponent Class Reference

Inheritance diagram for WeaponComponent:



Public Member Functions

- **WeaponComponent** (EntityId id)
- void **render** () override
- void **update** (const Input &inputSystem) override
- void **fixedUpdate** (const float &deltaTime) override
- std::string **name** () const override
- Component * **build** (EntityId entityId, const Components::component *component) override
- void **initialize** (EntityObject &entityParent) override
- void **shoot** (const TransformComponent &transform)

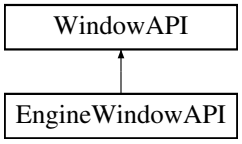
Additional Inherited Members

The documentation for this class was generated from the following files:

- Game/Components/WeaponComponent.hpp
- Game/Components/WeaponComponent.cpp

16.137 WindowAPI Class Reference

Inheritance diagram for WindowAPI:



Public Member Functions

- virtual void **initWindow** (int SCREEN_WIDTH, int SCREEN_HEIGHT) const =0
- virtual void **closeWindow** () const =0
- virtual SDL_Renderer * **getRenderer** () const =0

The documentation for this class was generated from the following file:

- API/Engine/WindowAPI.hpp

16.138 wl_buffer_listener Struct Reference

Public Attributes

- void(∗ release)(void ∗data, struct wl_buffer ∗wl_buffer)

16.138.1 Member Data Documentation

16.138.1.1 release

```
void(* wl_buffer_listener::release) (void *data, struct wl_buffer *wl_buffer)
```

compositor releases buffer

Sent when this wl_buffer is no longer used by the compositor. The client is now free to reuse or destroy this buffer and its backing storage.

If a client receives a release event before the frame callback requested in the same wl_surface.commit that attaches this wl_buffer to a surface, then the client is immediately free to reuse the buffer and its backing storage, and does not need a second buffer for the next surface content update. Typically this is possible, when the compositor maintains a copy of the wl_surface contents, e.g. as a GL texture. This is an important optimization for GL(ES) compositors with wl_shm clients.

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.139 wl_callback_listener Struct Reference

Public Attributes

- void(∗ done)(void ∗data, struct wl_callback ∗wl_callback, uint32_t callback_data)

16.139.1 Member Data Documentation

16.139.1.1 done

```
void(* wl_callback_listener::done) (void *data, struct wl_callback *wl_callback, uint32_↩
t callback_data)
```

done event

Notify the client when the related request is done.

Parameters

callback_data	request-specific data for the callback

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.140 wl_data_device_listener Struct Reference

Public Attributes

- void(∗ data_offer)(void ∗data, struct wl_data_device ∗wl_data_device, struct wl_data_offer ∗id)
- void(∗ enter)(void ∗data, struct wl_data_device ∗wl_data_device, uint32_t serial, struct wl_surface ∗surface, wl_fixed_t x, wl_fixed_t y, struct wl_data_offer ∗id)
- void(∗ leave)(void ∗data, struct wl_data_device ∗wl_data_device)
- void(∗ motion)(void ∗data, struct wl_data_device ∗wl_data_device, uint32_t time, wl_fixed_t x, wl_fixed_t y)
- void(∗ drop)(void ∗data, struct wl_data_device ∗wl_data_device)
- void(∗ selection)(void ∗data, struct wl_data_device ∗wl_data_device, struct wl_data_offer ∗id)

16.140.1 Member Data Documentation

16.140.1.1 data_offer

```
void(* wl_data_device_listener::data_offer) (void *data, struct wl_data_device *wl_data_device,
struct wl_data_offer *id)
```

introduce a new wl_data_offer

The data_offer event introduces a new wl_data_offer object, which will subsequently be used in either the data_↩ device.enter event (for drag-and-drop) or the data_device.selection event (for selections). Immediately following the data_device_data_offer event, the new data_offer object will send out data_offer.offer events to describe the mime types it offers.

Parameters

id	the new data_offer object

16.140.1.2 drop

```
void(* wl_data_device_listener::drop) (void *data, struct wl_data_device *wl_data_device)
```

end drag-and-drop session successfully

The event is sent when a drag-and-drop operation is ended because the implicit grab is removed.

The drag-and-drop destination is expected to honor the last action received through wl_data_offer.action, if the resulting action is "copy" or "move", the destination can still perform wl_data_offer.receive requests, and is expected to end all transfers with a wl_data_offer.finish request.

If the resulting action is "ask", the action will not be considered final. The drag-and-drop destination is expected to perform one last wl_data_offer.set_actions request, or wl_data_offer.destroy in order to cancel the operation.

16.140.1.3 enter

```
void(* wl_data_device_listener::enter) (void *data, struct wl_data_device *wl_data_device,
uint32_t serial, struct wl_surface *surface, wl_fixed_t x, wl_fixed_t y, struct wl_data_offer
*id)
```

initiate drag-and-drop session

This event is sent when an active drag-and-drop pointer enters a surface owned by the client. The position of the pointer at enter time is provided by the x and y arguments, in surface-local coordinates.

Parameters

serial	serial number of the enter event
surface	client surface entered
x	surface-local x coordinate
y	surface-local y coordinate
id	source data_offer object

16.140.1.4 leave

```
void(* wl_data_device_listener::leave) (void *data, struct wl_data_device *wl_data_device)
```

end drag-and-drop session

This event is sent when the drag-and-drop pointer leaves the surface and the session ends. The client must destroy the wl_data_offer introduced at enter time at this point.

16.140.1.5 motion

```
void(* wl_data_device_listener::motion) (void *data, struct wl_data_device *wl_data_device,
uint32_t time, wl_fixed_t x, wl_fixed_t y)
```

drag-and-drop session motion

This event is sent when the drag-and-drop pointer moves within the currently focused surface. The new position of the pointer is provided by the x and y arguments, in surface-local coordinates.

Parameters

time	timestamp with millisecond granularity
x	surface-local x coordinate
y	surface-local y coordinate

16.140.1.6 selection

```
void(* wl_data_device_listener::selection) (void *data, struct wl_data_device *wl_data_device,
struct wl_data_offer *id)
```

advertise new selection

The selection event is sent out to notify the client of a new wl_data_offer for the selection for this device. The data_device.data_offer and the data_offer.offer events are sent out immediately before this event to introduce the data offer object. The selection event is sent to a client immediately before receiving keyboard focus and when a new selection is set while the client has keyboard focus. The data_offer is valid until a new data_offer or NULL is received or until the client loses keyboard focus. The client must destroy the previous selection data_offer, if any, upon receiving this event.

Parameters

id	selection data_offer object

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.141 wl_data_offer_listener Struct Reference

Public Attributes

- void(∗ offer)(void ∗data, struct wl_data_offer ∗wl_data_offer, const char ∗mime_type)
- void(∗ source_actions)(void ∗data, struct wl_data_offer ∗wl_data_offer, uint32_t source_actions)
- void(∗ action)(void ∗data, struct wl_data_offer ∗wl_data_offer, uint32_t dnd_action)

16.141.1 Member Data Documentation

16.141.1.1 action

```
void(* wl_data_offer_listener::action) (void *data, struct wl_data_offer *wl_data_offer, uint32↩
_t dnd_action)
```

notify the selected action

This event indicates the action selected by the compositor after matching the source/destination side actions. Only one action (or none) will be offered here.

This event can be emitted multiple times during the drag-and-drop operation in response to destination side action changes through wl_data_offer.set_actions.

This event will no longer be emitted after wl_data_device.drop happened on the drag-and-drop destination, the client must honor the last action received, or the last preferred one set through wl_data_offer.set_actions when handling an "ask" action.

Compositors may also change the selected action on the fly, mainly in response to keyboard modifier changes during the drag-and-drop operation.

The most recent action received is always the valid one. Prior to receiving wl_data_device.drop, the chosen action may change (e.g. due to keyboard modifiers being pressed). At the time of receiving wl_data_device.drop the drag-and-drop destination must honor the last action received.

Action changes may still happen after wl_data_device.drop, especially on "ask" actions, where the drag-and-drop destination may choose another action afterwards. Action changes happening at this stage are always the result of inter-client negotiation, the compositor shall no longer be able to induce a different action.

Upon "ask" actions, it is expected that the drag-and-drop destination may potentially choose a different action and/or mime type, based on wl_data_offer.source_actions and finally chosen by the user (e.g. popping up a menu with the available options). The final wl_data_offer.set_actions and wl_data_offer.accept requests must happen before the call to wl_data_offer.finish.

Parameters

dnd_action	action selected by the compositor

Since

3

16.141.1.2 offer

```
void(* wl_data_offer_listener::offer) (void *data, struct wl_data_offer *wl_data_offer, const
char *mime_type)
```

advertise offered mime type

Sent immediately after creating the wl_data_offer object. One event per offered mime type.

Parameters

mime_type	offered mime type

16.141.1.3 source_actions

```
void(* wl_data_offer_listener::source_actions) (void *data, struct wl_data_offer *wl_data_↩
offer, uint32_t source_actions)
```

notify the source-side available actions

This event indicates the actions offered by the data source. It will be sent right after wl_data_device.enter, or anytime the source side changes its offered actions through wl_data_source.set_actions.

Parameters

source_actions	actions offered by the data source

Since

3

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.142 wl_data_source_listener Struct Reference

Public Attributes

- void(∗ target)(void ∗data, struct wl_data_source ∗wl_data_source, const char ∗mime_type)
- void(∗ send)(void ∗data, struct wl_data_source ∗wl_data_source, const char ∗mime_type, int32_t fd)
- void(∗ cancelled)(void ∗data, struct wl_data_source ∗wl_data_source)
- void(∗ dnd_drop_performed)(void ∗data, struct wl_data_source ∗wl_data_source)
- void(∗ dnd_finished)(void ∗data, struct wl_data_source ∗wl_data_source)
- void(∗ action)(void ∗data, struct wl_data_source ∗wl_data_source, uint32_t dnd_action)

16.142.1 Member Data Documentation

16.142.1.1 action

```
void(* wl_data_source_listener::action) (void *data, struct wl_data_source *wl_data_source,
uint32_t dnd_action)
```

notify the selected action

This event indicates the action selected by the compositor after matching the source/destination side actions. Only one action (or none) will be offered here.

This event can be emitted multiple times during the drag-and-drop operation, mainly in response to destination side changes through wl_data_offer.set_actions, and as the data device enters/leaves surfaces.

It is only possible to receive this event after wl_data_source.dnd_drop_performed if the drag-and-drop operation ended in an "ask" action, in which case the final wl_data_source.action event will happen immediately before wl_↩ data_source.dnd_finished.

Compositors may also change the selected action on the fly, mainly in response to keyboard modifier changes during the drag-and-drop operation.

The most recent action received is always the valid one. The chosen action may change alongside negotiation (e.g. an "ask" action can turn into a "move" operation), so the effects of the final action must always be applied in wl_data_offer.dnd_finished.

Clients can trigger cursor surface changes from this point, so they reflect the current action.

Parameters

dnd_action	action selected by the compositor

Since

3

16.142.1.2 cancelled

```
void(* wl_data_source_listener::cancelled) (void *data, struct wl_data_source *wl_data_source)
```

selection was cancelled

This data source is no longer valid. There are several reasons why this could happen:

- The data source has been replaced by another data source. - The drag-and-drop operation was performed, but the drop destination did not accept any of the mime types offered through wl_data_source.target. - The drag-and-drop operation was performed, but the drop destination did not select any of the actions present in the mask offered through wl_data_source.action. - The drag-and-drop operation was performed but didn't happen over a surface. - The compositor cancelled the drag-and-drop operation (e.g. compositor dependent timeouts to avoid stale drag-and-drop transfers).

The client should clean up and destroy this data source.

For objects of version 2 or older, wl_data_source.cancelled will only be emitted if the data source was replaced by another data source.

16.142.1.3 dnd_drop_performed

```
void(* wl_data_source_listener::dnd_drop_performed) (void *data, struct wl_data_source *wl_↩
data_source)
```

the drag-and-drop operation physically finished

The user performed the drop action. This event does not indicate acceptance, wl_data_source.cancelled may still be emitted afterwards if the drop destination does not accept any mime type.

However, this event might however not be received if the compositor cancelled the drag-and-drop operation before this event could happen.

Note that the data_source may still be used in the future and should not be destroyed here.

Since

3

16.142.1.4 dnd_finished

```
void(* wl_data_source_listener::dnd_finished) (void *data, struct wl_data_source *wl_data_↩
source)
```

the drag-and-drop operation concluded

The drop destination finished interoperating with this data source, so the client is now free to destroy this data source and free all associated data.

If the action used to perform the operation was "move", the source can now delete the transferred data.

Since

3

16.142.1.5 send

```
void(* wl_data_source_listener::send) (void *data, struct wl_data_source *wl_data_source, const
char *mime_type, int32_t fd)
```

send the data

Request for data from the client. Send the data as the specified mime type over the passed file descriptor, then close it.

Parameters

mime_type	mime type for the data
fd	file descriptor for the data

16.142.1.6 target

```
void(* wl_data_source_listener::target) (void *data, struct wl_data_source *wl_data_source,
const char *mime_type)
```

a target accepts an offered mime type

Sent when a target accepts pointer_focus or motion events. If a target does not accept any of the offered types, type is NULL.

Used for feedback during drag-and-drop.

Parameters

mime_type	mime type accepted by the target

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.143 wl_display_listener Struct Reference

Public Attributes

- void(∗ error)(void ∗data, struct wl_display ∗wl_display, void ∗object_id, uint32_t code, const char ∗message)
- void(∗ delete_id)(void ∗data, struct wl_display ∗wl_display, uint32_t id)

16.143.1 Member Data Documentation

16.143.1.1 delete_id

```
void(* wl_display_listener::delete_id) (void *data, struct wl_display *wl_display, uint32_t id)
```

acknowledge object ID deletion

This event is used internally by the object ID management logic. When a client deletes an object, the server will send this event to acknowledge that it has seen the delete request. When the client receives this event, it will know that it can safely reuse the object ID.

Parameters

id	deleted object ID

16.143.1.2 error

```
void(* wl_display_listener::error) (void *data, struct wl_display *wl_display, void *object_id,
uint32_t code, const char *message)
```

fatal error event

The error event is sent out when a fatal (non-recoverable) error has occurred. The object_id argument is the object where the error occurred, most often in response to a request to that object. The code identifies the error and is defined by the object interface. As such, each interface defines its own set of error codes. The message is a brief description of the error, for (debugging) convenience.

Parameters

object_id	object where the error occurred
code	error code
message	error description

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.144 wl_keyboard_listener Struct Reference

Public Attributes

- void(* keymap)(void *data, struct wl_keyboard *wl_keyboard, uint32_t format, int32_t fd, uint32_t size)
- void(* enter)(void *data, struct wl_keyboard *wl_keyboard, uint32_t serial, struct wl_surface *surface, struct wl_array *keys)
- void(* leave)(void *data, struct wl_keyboard *wl_keyboard, uint32_t serial, struct wl_surface *surface)
- void(* key)(void *data, struct wl_keyboard *wl_keyboard, uint32_t serial, uint32_t time, uint32_t key, uint32_t state)
- void(* modifiers)(void *data, struct wl_keyboard *wl_keyboard, uint32_t serial, uint32_t mods_depressed, uint32_t mods_latched, uint32_t mods_locked, uint32_t group)
- void(* repeat_info)(void *data, struct wl_keyboard *wl_keyboard, int32_t rate, int32_t delay)

16.144.1 Member Data Documentation

16.144.1.1 enter

```
void(* wl_keyboard_listener::enter) (void *data, struct wl_keyboard *wl_keyboard, uint32_t serial, struct wl_surface *surface, struct wl_array *keys)
```

enter event

Notification that this seat's keyboard focus is on a certain surface.

Parameters

serial	serial number of the enter event
surface	surface gaining keyboard focus
keys	the currently pressed keys

16.144.1.2 key

```
void(* wl_keyboard_listener::key) (void *data, struct wl_keyboard *wl_keyboard, uint32_t serial,
uint32_t time, uint32_t key, uint32_t state)
```

key event

A key was pressed or released. The time argument is a timestamp with millisecond granularity, with an undefined base.

Parameters

serial	serial number of the key event
time	timestamp with millisecond granularity
key	key that produced the event
state	physical state of the key

16.144.1.3 keymap

```
void(* wl_keyboard_listener::keymap) (void *data, struct wl_keyboard *wl_keyboard, uint32_↩
t format, int32_t fd, uint32_t size)
```

keyboard mapping

This event provides a file descriptor to the client which can be memory-mapped to provide a keyboard mapping description.

Parameters

format	keymap format
fd	keymap file descriptor
size	keymap size, in bytes

16.144.1.4 leave

```
void(* wl_keyboard_listener::leave) (void *data, struct wl_keyboard *wl_keyboard, uint32_↩
t serial, struct wl_surface *surface)
```

leave event

Notification that this seat's keyboard focus is no longer on a certain surface.

The leave notification is sent before the enter notification for the new focus.

Parameters

serial	serial number of the leave event
surface	surface that lost keyboard focus

16.144.1.5 modifiers

```
void(* wl_keyboard_listener::modifiers) (void *data, struct wl_keyboard *wl_keyboard, uint32_t
serial, uint32_t mods_depressed, uint32_t mods_latched, uint32_t mods_locked, uint32_t group)
```

modifier and group state

Notifies clients that the modifier and/or group state has changed, and it should update its local state.

Parameters

serial	serial number of the modifiers event
mods_depressed	depressed modifiers
mods_latched	latched modifiers
mods_locked	locked modifiers
group	keyboard layout

16.144.1.6 repeat_info

```
void(* wl_keyboard_listener::repeat_info) (void *data, struct wl_keyboard *wl_keyboard, int32↩
_t rate, int32_t delay)
```

repeat rate and delay

Informs the client about the keyboard's repeat rate and delay.

This event is sent as soon as the wl_keyboard object has been created, and is guaranteed to be received by the client before any key press event.

Negative values for either rate or delay are illegal. A rate of zero will disable any repeating (regardless of the value of delay).

This event can be sent later on as well with a new value if necessary, so clients should continue listening for the event past the creation of wl_keyboard.

Parameters

rate	the rate of repeating keys in characters per second
delay	delay in milliseconds since key down until repeating starts

Since

4

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.145 wl_output_listener Struct Reference

Public Attributes

- void(∗ geometry)(void ∗data, struct wl_output ∗wl_output, int32_t x, int32_t y, int32_t physical_width, int32_t physical_height, int32_t subpixel, const char ∗make, const char ∗model, int32_t transform)
- void(∗ mode)(void ∗data, struct wl_output ∗wl_output, uint32_t flags, int32_t width, int32_t height, int32_t refresh)
- void(∗ done)(void ∗data, struct wl_output ∗wl_output)
- void(∗ scale)(void ∗data, struct wl_output ∗wl_output, int32_t factor)

16.145.1 Member Data Documentation

16.145.1.1 done

```
void(* wl_output_listener::done) (void *data, struct wl_output *wl_output)
```

sent all information about output

This event is sent after all other properties have been sent after binding to the output object and after any other property changes done after that. This allows changes to the output properties to be seen as atomic, even if they happen via multiple events.

Since

2

16.145.1.2 geometry

```
void(* wl_output_listener::geometry) (void *data, struct wl_output *wl_output, int32_t x, int32↩
_t y, int32_t physical_width, int32_t physical_height, int32_t subpixel, const char *make,
const char *model, int32_t transform)
```

properties of the output

The geometry event describes geometric properties of the output. The event is sent when binding to the output object and whenever any of the properties change.

Parameters

x	x position within the global compositor space
y	y position within the global compositor space
physical_width	width in millimeters of the output
physical_height	height in millimeters of the output
subpixel	subpixel orientation of the output
make	textual description of the manufacturer
model	textual description of the model
transform	transform that maps framebuffer to output

16.145.1.3 mode

```
void(* wl_output_listener::mode) (void *data, struct wl_output *wl_output, uint32_t flags,
int32_t width, int32_t height, int32_t refresh)
```

advertise available modes for the output

The mode event describes an available mode for the output.

The event is sent when binding to the output object and there will always be one mode, the current mode. The event is sent again if an output changes mode, for the mode that is now current. In other words, the current mode is always the last mode that was received with the current flag set.

The size of a mode is given in physical hardware units of the output device. This is not necessarily the same as the output size in the global compositor space. For instance, the output may be scaled, as described in wl_output.scale, or transformed, as described in wl_output.transform.

Parameters

flags	bitfield of mode flags
width	width of the mode in hardware units
height	height of the mode in hardware units
refresh	vertical refresh rate in mHz

16.145.1.4 scale

```
void(* wl_output_listener::scale) (void *data, struct wl_output *wl_output, int32_t factor)
```

output scaling properties

This event contains scaling geometry information that is not in the geometry event. It may be sent after binding the output object or if the output scale changes later. If it is not sent, the client should assume a scale of 1.

A scale larger than 1 means that the compositor will automatically scale surface buffers by this amount when rendering. This is used for very high resolution displays where applications rendering at the native resolution would be too small to be legible.

It is intended that scaling aware clients track the current output of a surface, and if it is on a scaled output it should use wl_surface.set_buffer_scale with the scale of the output. That way the compositor can avoid scaling the surface, and the client can supply a higher detail image.

Parameters

factor	scaling factor of output

Since

2

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.146 wl_pointer_listener Struct Reference

Public Attributes

- void(∗ enter)(void ∗data, struct wl_pointer ∗wl_pointer, uint32_t serial, struct wl_surface ∗surface, wl_fixed↩ _t surface_x, wl_fixed_t surface_y)
- void(∗ leave)(void ∗data, struct wl_pointer ∗wl_pointer, uint32_t serial, struct wl_surface ∗surface)
- void(∗ motion)(void ∗data, struct wl_pointer ∗wl_pointer, uint32_t time, wl_fixed_t surface_x, wl_fixed_↩ t surface_y)
- void(∗ button)(void ∗data, struct wl_pointer ∗wl_pointer, uint32_t serial, uint32_t time, uint32_t button, uint32_t state)
- void(∗ axis)(void ∗data, struct wl_pointer ∗wl_pointer, uint32_t time, uint32_t axis, wl_fixed_t value)
- void(∗ frame)(void ∗data, struct wl_pointer ∗wl_pointer)
- void(∗ axis_source)(void ∗data, struct wl_pointer ∗wl_pointer, uint32_t axis_source)
- void(∗ axis_stop)(void ∗data, struct wl_pointer ∗wl_pointer, uint32_t time, uint32_t axis)
- void(∗ axis_discrete)(void ∗data, struct wl_pointer ∗wl_pointer, uint32_t axis, int32_t discrete)

16.146.1 Member Data Documentation

16.146.1.1 axis

```
void(* wl_pointer_listener::axis) (void *data, struct wl_pointer *wl_pointer, uint32_t time,
uint32_t axis, wl_fixed_t value)
```

axis event

Scroll and other axis notifications.

For scroll events (vertical and horizontal scroll axes), the value parameter is the length of a vector along the specified axis in a coordinate space identical to those of motion events, representing a relative movement along the specified axis.

For devices that support movements non-parallel to axes multiple axis events will be emitted.

When applicable, for example for touch pads, the server can choose to emit scroll events where the motion vector is equivalent to a motion event vector.

When applicable, a client can transform its content relative to the scroll distance.

Parameters

time	timestamp with millisecond granularity
axis	axis type
value	length of vector in surface-local coordinate space

16.146.1.2 axis_discrete

```
void(* wl_pointer_listener::axis_discrete) (void *data, struct wl_pointer *wl_pointer, uint32_t axis, int32_t discrete)
```

axis click event

Discrete step information for scroll and other axes.

This event carries the axis value of the wl_pointer.axis event in discrete steps (e.g. mouse wheel clicks).

This event does not occur on its own, it is coupled with a wl_pointer.axis event that represents this axis value on a continuous scale. The protocol guarantees that each axis_discrete event is always followed by exactly one axis event with the same axis number within the same wl_pointer.frame. Note that the protocol allows for other events to occur between the axis_discrete and its coupled axis event, including other axis_discrete or axis events.

This event is optional; continuous scrolling devices like two-finger scrolling on touchpads do not have discrete steps and do not generate this event.

The discrete value carries the directional information. e.g. a value of -2 is two steps towards the negative direction of this axis.

The axis number is identical to the axis number in the associated axis event.

The order of wl_pointer.axis_discrete and wl_pointer.axis_source is not guaranteed.

Parameters

axis	axis type
discrete	number of steps

Since

5

16.146.1.3 axis_source

```
void(* wl_pointer_listener::axis_source) (void *data, struct wl_pointer *wl_pointer, uint32_t axis_source)
```

axis source event

Source information for scroll and other axes.

This event does not occur on its own. It is sent before a wl_pointer.frame event and carries the source information for all events within that frame.

The source specifies how this event was generated. If the source is wl_pointer.axis_source.finger, a wl_pointer.axis_stop event will be sent when the user lifts the finger off the device.

If the source is wl_pointer.axis_source.wheel, wl_pointer.axis_source.wheel_tilt or wl_pointer.axis_source.continuous, a wl_pointer.axis_stop event may or may not be sent. Whether a compositor sends an axis_stop event for these sources is hardware-specific and implementation-dependent; clients must not rely on receiving an axis_stop event for these scroll sources and should treat scroll sequences from these scroll sources as unterminated by default.

This event is optional. If the source is unknown for a particular axis event sequence, no event is sent. Only one wl_pointer.axis_source event is permitted per frame.

The order of wl_pointer.axis_discrete and wl_pointer.axis_source is not guaranteed.

Parameters

axis_source	source of the axis event

Since

5

16.146.1.4 axis_stop

```
void(* wl_pointer_listener::axis_stop) (void *data, struct wl_pointer *wl_pointer, uint32_t time, uint32_t axis)
```

axis stop event

Stop notification for scroll and other axes.

For some wl_pointer.axis_source types, a wl_pointer.axis_stop event is sent to notify a client that the axis sequence has terminated. This enables the client to implement kinetic scrolling. See the wl_pointer.axis_source documentation for information on when this event may be generated.

Any wl_pointer.axis events with the same axis_source after this event should be considered as the start of a new axis motion.

The timestamp is to be interpreted identical to the timestamp in the wl_pointer.axis event. The timestamp value may be the same as a preceding wl_pointer.axis event.

Parameters

time	timestamp with millisecond granularity
axis	the axis stopped with this event

Since

5

16.146.1.5 button

```
void(* wl_pointer_listener::button) (void *data, struct wl_pointer *wl_pointer, uint32_t serial,
uint32_t time, uint32_t button, uint32_t state)
```

pointer button event

Mouse button click and release notifications.

The location of the click is given by the last motion or enter event. The time argument is a timestamp with millisecond granularity, with an undefined base.

The button is a button code as defined in the Linux kernel's linux/input-event-codes.h header file, e.g. BTN_LEFT.

Any 16-bit button code value is reserved for future additions to the kernel's event code list. All other button codes above 0xFFFF are currently undefined but may be used in future versions of this protocol.

Parameters

serial	serial number of the button event
time	timestamp with millisecond granularity
button	button that produced the event
state	physical state of the button

16.146.1.6 enter

```
void(* wl_pointer_listener::enter) (void *data, struct wl_pointer *wl_pointer, uint32_t serial,
struct wl_surface *surface, wl_fixed_t surface_x, wl_fixed_t surface_y)
```

enter event

Notification that this seat's pointer is focused on a certain surface.

When a seat's focus enters a surface, the pointer image is undefined and a client should respond to this event by setting an appropriate pointer image with the set_cursor request.

Parameters

serial	serial number of the enter event
surface	surface entered by the pointer
surface_x	surface-local x coordinate
surface_y	surface-local y coordinate

16.146.1.7 frame

```
void(* wl_pointer_listener::frame) (void *data, struct wl_pointer *wl_pointer)
```

end of a pointer event sequence

Indicates the end of a set of events that logically belong together. A client is expected to accumulate the data in all events within the frame before proceeding.

All wl_pointer events before a wl_pointer.frame event belong logically together. For example, in a diagonal scroll motion the compositor will send an optional wl_pointer.axis_source event, two wl_pointer.axis events (horizontal and vertical) and finally a wl_pointer.frame event. The client may use this information to calculate a diagonal vector for scrolling.

When multiple wl_pointer.axis events occur within the same frame, the motion vector is the combined motion of all events. When a wl_pointer.axis and a wl_pointer.axis_stop event occur within the same frame, this indicates that axis movement in one axis has stopped but continues in the other axis. When multiple wl_pointer.axis_stop events occur within the same frame, this indicates that these axes stopped in the same instance.

A wl_pointer.frame event is sent for every logical event group, even if the group only contains a single wl_pointer event. Specifically, a client may get a sequence: motion, frame, button, frame, axis, frame, axis_stop, frame.

The wl_pointer.enter and wl_pointer.leave events are logical events generated by the compositor and not the hardware. These events are also grouped by a wl_pointer.frame. When a pointer moves from one surface to another, a compositor should group the wl_pointer.leave event within the same wl_pointer.frame. However, a client must not rely on wl_pointer.leave and wl_pointer.enter being in the same wl_pointer.frame. Compositor-specific policies may require the wl_pointer.leave and wl_pointer.enter event being split across multiple wl_pointer.frame groups.

Since

5

16.146.1.8 leave

```
void(* wl_pointer_listener::leave) (void *data, struct wl_pointer *wl_pointer, uint32_t serial,
struct wl_surface *surface)
```

leave event

Notification that this seat's pointer is no longer focused on a certain surface.

The leave notification is sent before the enter notification for the new focus.

Parameters

serial	serial number of the leave event
surface	surface left by the pointer

16.146.1.9 motion

```
void(* wl_pointer_listener::motion) (void *data, struct wl_pointer *wl_pointer, uint32_t time,
wl_fixed_t surface_x, wl_fixed_t surface_y)
```

pointer motion event

Notification of pointer location change. The arguments surface_x and surface_y are the location relative to the focused surface.

Parameters

time	timestamp with millisecond granularity
surface_x	surface-local x coordinate
surface_y	surface-local y coordinate

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.147 wl_registry_listener Struct Reference

Public Attributes

- void(∗ global)(void ∗data, struct wl_registry ∗wl_registry, uint32_t name, const char ∗interface, uint32_t version)
- void(∗ global_remove)(void ∗data, struct wl_registry ∗wl_registry, uint32_t name)

16.147.1 Member Data Documentation

16.147.1.1 global

```
void(* wl_registry_listener::global) (void *data, struct wl_registry *wl_registry, uint32_t
name, const char *interface, uint32_t version)
```

announce global object

Notify the client of global objects.

The event notifies the client that a global object with the given name is now available, and it implements the given version of the given interface.

Parameters

name	numeric name of the global object
interface	interface implemented by the object
version	interface version

16.147.1.2 global_remove

```
void(* wl_registry_listener::global_remove) (void *data, struct wl_registry *wl_registry, uint32↩
_t name)
```

announce removal of global object

Notify the client of removed global objects.

This event notifies the client that the global identified by name is no longer available. If the client bound to the global using the bind request, the client should now destroy that object.

The object remains valid and requests to the object will be ignored until the client destroys it, to avoid races between the global going away and a client sending a request to it.

Parameters

name	numeric name of the global object

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.148 wl_seat_listener Struct Reference

Public Attributes

- void(∗ capabilities)(void ∗data, struct wl_seat ∗wl_seat, uint32_t capabilities)
- void(∗ name)(void ∗data, struct wl_seat ∗wl_seat, const char ∗name)

16.148.1 Member Data Documentation

16.148.1.1 capabilities

```
void(* wl_seat_listener::capabilities) (void *data, struct wl_seat *wl_seat, uint32_t capabilities)
```

seat capabilities changed

This is emitted whenever a seat gains or loses the pointer, keyboard or touch capabilities. The argument is a capability enum containing the complete set of capabilities this seat has.

When the pointer capability is added, a client may create a wl_pointer object using the wl_seat.get_pointer request. This object will receive pointer events until the capability is removed in the future.

When the pointer capability is removed, a client should destroy the wl_pointer objects associated with the seat where the capability was removed, using the wl_pointer.release request. No further pointer events will be received on these objects.

In some compositors, if a seat regains the pointer capability and a client has a previously obtained wl_pointer object of version 4 or less, that object may start sending pointer events again. This behavior is considered a misinterpretation of the intended behavior and must not be relied upon by the client. wl_pointer objects of version 5 or later must not send events if created before the most recent event notifying the client of an added pointer capability.

The above behavior also applies to wl_keyboard and wl_touch with the keyboard and touch capabilities, respectively.

Parameters

capabilities	capabilities of the seat

16.148.1.2 name

```
void(* wl_seat_listener::name) (void *data, struct wl_seat *wl_seat, const char *name)
```

unique identifier for this seat

In a multiseat configuration this can be used by the client to help identify which physical devices the seat represents. Based on the seat configuration used by the compositor.

Parameters

name	seat identifier

Since

2

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.149 wl_shell_surface_listener Struct Reference

Public Attributes

- void(∗ ping)(void ∗data, struct wl_shell_surface ∗wl_shell_surface, uint32_t serial)
- void(∗ configure)(void ∗data, struct wl_shell_surface ∗wl_shell_surface, uint32_t edges, int32_t width, int32_t height)
- void(∗ popup_done)(void ∗data, struct wl_shell_surface ∗wl_shell_surface)

16.149.1 Member Data Documentation

16.149.1.1 configure

```
void(* wl_shell_surface_listener::configure) (void *data, struct wl_shell_surface *wl_shell_surface, uint32_t edges, int32_t width, int32_t height)
```

suggest resize

The configure event asks the client to resize its surface.

The size is a hint, in the sense that the client is free to ignore it if it doesn't resize, pick a smaller size (to satisfy aspect ratio or resize in steps of NxM pixels).

The edges parameter provides a hint about how the surface was resized. The client may use this information to decide how to adjust its content to the new size (e.g. a scrolling area might adjust its content position to leave the viewable content unmoved).

The client is free to dismiss all but the last configure event it received.

The width and height arguments specify the size of the window in surface-local coordinates.

Parameters

edges	how the surface was resized
width	new width of the surface
height	new height of the surface

16.149.1.2 ping

```
void(* wl_shell_surface_listener::ping) (void *data, struct wl_shell_surface *wl_shell_surface, uint32_t serial)
```

ping client

Ping a client to check if it is receiving events and sending requests. A client is expected to reply with a pong request.

Parameters

serial	serial number of the ping

16.149.1.3 popup_done

```
void(* wl_shell_surface_listener::popup_done) (void *data, struct wl_shell_surface *wl_shell_↩
surface)
```

popup interaction is done

The popup_done event is sent out when a popup grab is broken, that is, when the user clicks a surface that doesn't belong to the client owning the popup surface.

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.150 wl_shm_listener Struct Reference

Public Attributes

- void(∗ format)(void ∗data, struct wl_shm ∗wl_shm, uint32_t format)

16.150.1 Member Data Documentation

16.150.1.1 format

```
void(* wl_shm_listener::format) (void *data, struct wl_shm *wl_shm, uint32_t format)
```

pixel format description

Informs the client about a valid pixel format that can be used for buffers. Known formats include argb8888 and xrgb8888.

Parameters

format	buffer pixel format

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.151 wl_surface_listener Struct Reference

Public Attributes

- void(∗ enter)(void ∗data, struct wl_surface ∗wl_surface, struct wl_output ∗output)
- void(∗ leave)(void ∗data, struct wl_surface ∗wl_surface, struct wl_output ∗output)

16.151.1 Member Data Documentation

16.151.1.1 enter

```
void(* wl_surface_listener::enter) (void *data, struct wl_surface *wl_surface, struct wl_output
*output)
```

surface enters an output

This is emitted whenever a surface's creation, movement, or resizing results in some part of it being within the scanout region of an output.

Note that a surface may be overlapping with zero or more outputs.

Parameters

output	output entered by the surface

16.151.1.2 leave

```
void(* wl_surface_listener::leave) (void *data, struct wl_surface *wl_surface, struct wl_output
*output)
```

surface leaves an output

This is emitted whenever a surface's creation, movement, or resizing results in it no longer having any part of it within the scanout region of an output.

Parameters

output	output left by the surface

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.152 wl_touch_listener Struct Reference

Public Attributes

- void(∗ down)(void ∗data, struct wl_touch ∗wl_touch, uint32_t serial, uint32_t time, struct wl_surface ∗surface, int32_t id, wl_fixed_t x, wl_fixed_t y)
- void(∗ up)(void ∗data, struct wl_touch ∗wl_touch, uint32_t serial, uint32_t time, int32_t id)
- void(∗ motion)(void ∗data, struct wl_touch ∗wl_touch, uint32_t time, int32_t id, wl_fixed_t x, wl_fixed_t y)
- void(∗ frame)(void ∗data, struct wl_touch ∗wl_touch)
- void(∗ cancel)(void ∗data, struct wl_touch ∗wl_touch)
- void(∗ shape)(void ∗data, struct wl_touch ∗wl_touch, int32_t id, wl_fixed_t major, wl_fixed_t minor)
- void(∗ orientation)(void ∗data, struct wl_touch ∗wl_touch, int32_t id, wl_fixed_t orientation)

16.152.1 Member Data Documentation

16.152.1.1 cancel

```
void(* wl_touch_listener::cancel) (void *data, struct wl_touch *wl_touch)
```

touch session cancelled

Sent if the compositor decides the touch stream is a global gesture. No further events are sent to the clients from that particular gesture. Touch cancellation applies to all touch points currently active on this client's surface. The client is responsible for finalizing the touch points, future touch points on this surface may reuse the touch point ID.

16.152.1.2 down

```
void(* wl_touch_listener::down) (void *data, struct wl_touch *wl_touch, uint32_t serial, uint32_t time, struct wl_surface *surface, int32_t id, wl_fixed_t x, wl_fixed_t y)
```

touch down event and beginning of a touch sequence

A new touch point has appeared on the surface. This touch point is assigned a unique ID. Future events from this touch point reference this ID. The ID ceases to be valid after a touch up event and may be reused in the future.

Parameters

serial	serial number of the touch down event
time	timestamp with millisecond granularity
surface	surface touched
id	the unique ID of this touch point
x	surface-local x coordinate
y	surface-local y coordinate

16.152.1.3 frame

```
void(* wl_touch_listener::frame) (void *data, struct wl_touch *wl_touch)
```

end of touch frame event

Indicates the end of a set of events that logically belong together. A client is expected to accumulate the data in all events within the frame before proceeding.

A wl_touch.frame terminates at least one event but otherwise no guarantee is provided about the set of events within a frame. A client must assume that any state not updated in a frame is unchanged from the previously known state.

16.152.1.4 motion

```
void(* wl_touch_listener::motion) (void *data, struct wl_touch *wl_touch, uint32_t time, int32↩
_t id, wl_fixed_t x, wl_fixed_t y)
```

update of touch point coordinates

A touch point has changed coordinates.

Parameters

time	timestamp with millisecond granularity
id	the unique ID of this touch point
x	surface-local x coordinate
y	surface-local y coordinate

16.152.1.5 orientation

```
void(* wl_touch_listener::orientation) (void *data, struct wl_touch *wl_touch, int32_t id, wl↩
_fixed_t orientation)
```

update orientation of touch point

Sent when a touchpoint has changed its orientation.

This event does not occur on its own. It is sent before a wl_touch.frame event and carries the new shape information for any previously reported, or new touch points of that frame.

Other events describing the touch point such as wl_touch.down, wl_touch.motion or wl_touch.shape may be sent within the same wl_touch.frame. A client should treat these events as a single logical touch point update. The order of wl_touch.shape, wl_touch.orientation and wl_touch.motion is not guaranteed. A wl_touch.down event is guaranteed to occur before the first wl_touch.orientation event for this touch ID but both events may occur within the same wl_touch.frame.

The orientation describes the clockwise angle of a touchpoint's major axis to the positive surface y-axis and is normalized to the -180 to +180 degree range. The granularity of orientation depends on the touch device, some devices only support binary rotation values between 0 and 90 degrees.

This event is only sent by the compositor if the touch device supports orientation reports.

Parameters

id	the unique ID of this touch point
orientation	angle between major axis and positive surface y-axis in degrees

Since

6

16.152.1.6 shape

```
void(* wl_touch_listener::shape) (void *data, struct wl_touch *wl_touch, int32_t id, wl_fixed↩
_t major, wl_fixed_t minor)
```

update shape of touch point

Sent when a touchpoint has changed its shape.

This event does not occur on its own. It is sent before a wl_touch.frame event and carries the new shape information for any previously reported, or new touch points of that frame.

Other events describing the touch point such as wl_touch.down, wl_touch.motion or wl_touch.orientation may be sent within the same wl_touch.frame. A client should treat these events as a single logical touch point update. The order of wl_touch.shape, wl_touch.orientation and wl_touch.motion is not guaranteed. A wl_touch.down event is guaranteed to occur before the first wl_touch.shape event for this touch ID but both events may occur within the same wl_touch.frame.

A touchpoint shape is approximated by an ellipse through the major and minor axis length. The major axis length describes the longer diameter of the ellipse, while the minor axis length describes the shorter diameter. Major and minor are orthogonal and both are specified in surface-local coordinates. The center of the ellipse is always at the touchpoint location as reported by wl_touch.down or wl_touch.move.

This event is only sent by the compositor if the touch device supports shape reports. The client has to make reasonable assumptions about the shape if it did not receive this event.

Parameters

id	the unique ID of this touch point
major	length of the major axis in surface-local coordinates
minor	length of the minor axis in surface-local coordinates

Since

6

16.152.1.7 up

```
void(* wl_touch_listener::up) (void *data, struct wl_touch *wl_touch, uint32_t serial, uint32_t time, int32_t id)
```

end of a touch event sequence

The touch point has disappeared. No further events will be sent for this touch point and the touch point's ID is released and may be reused in a future touch down event.

Parameters

serial	serial number of the touch up event
time	timestamp with millisecond granularity
id	the unique ID of this touch point

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/wayland-client-protocol.h

16.153 xdg_popup_listener Struct Reference

Public Attributes

- void(∗ configure)(void ∗data, struct xdg_popup ∗xdg_popup, int32_t x, int32_t y, int32_t width, int32_t height)
- void(∗ popup_done)(void ∗data, struct xdg_popup ∗xdg_popup)

16.153.1 Member Data Documentation

16.153.1.1 configure

```
void(* xdg_popup_listener::configure) (void *data, struct xdg_popup *xdg_popup, int32_t x, int32_t y, int32_t width, int32_t height)
```

configure the popup surface

This event asks the popup surface to configure itself given the configuration. The configured state should not be applied immediately. See xdg_surface.configure for details.

The x and y arguments represent the position the popup was placed at given the xdg_positioner rule, relative to the upper left corner of the window geometry of the parent surface.

Parameters

x	x position relative to parent surface window geometry
y	y position relative to parent surface window geometry
width	window geometry width
height	window geometry height

16.153.1.2 popup_done

```
void(* xdg_popup_listener::popup_done) (void *data, struct xdg_popup *xdg_popup)
```

popup interaction is done

The popup_done event is sent out when a popup is dismissed by the compositor. The client should destroy the xdg_popup object at this point.

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/xdg-shell-client-protocol.h

16.154 xdg_surface_listener Struct Reference

Public Attributes

- void(∗ configure)(void ∗data, struct xdg_surface ∗xdg_surface, uint32_t serial)

16.154.1 Member Data Documentation

16.154.1.1 configure

```
void(* xdg_surface_listener::configure) (void *data, struct xdg_surface *xdg_surface, uint32_t
serial)
```

suggest a surface change

The configure event marks the end of a configure sequence. A configure sequence is a set of one or more events configuring the state of the xdg_surface, including the final xdg_surface.configure event.

Where applicable, xdg_surface surface roles will during a configure sequence extend this event as a latched state sent as events before the xdg_surface.configure event. Such events should be considered to make up a set of atomically applied configuration states, where the xdg_surface.configure commits the accumulated state.

Clients should arrange their surface for the new states, and then send an ack_configure request with the serial sent in this configure event at some point before committing the new surface.

If the client receives multiple configure events before it can respond to one, it is free to discard all but the last event it received.

Parameters

serial	serial of the configure event

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/xdg-shell-client-protocol.h

16.155 xdg_toplevel_listener Struct Reference

Public Attributes

- void(∗ configure)(void ∗data, struct xdg_toplevel ∗xdg_toplevel, int32_t width, int32_t height, struct wl_array ∗states)
- void(∗ close)(void ∗data, struct xdg_toplevel ∗xdg_toplevel)

16.155.1 Member Data Documentation

16.155.1.1 close

```
void(* xdg_toplevel_listener::close) (void *data, struct xdg_toplevel *xdg_toplevel)
```

surface wants to be closed

The close event is sent by the compositor when the user wants the surface to be closed. This should be equivalent to the user clicking the close button in client-side decorations, if your application has any.

This is only a request that the user intends to close the window. The client may choose to ignore this request, or show a dialog to ask the user to save their data, etc.

16.155.1.2 configure

```
void(* xdg_toplevel_listener::configure) (void *data, struct xdg_toplevel *xdg_toplevel, int32↩
_t width, int32_t height, struct wl_array *states)
```

suggest a surface change

This configure event asks the client to resize its toplevel surface or to change its state. The configured state should not be applied immediately. See xdg_surface.configure for details.

The width and height arguments specify a hint to the window about how its surface should be resized in window geometry coordinates. See set_window_geometry.

If the width or height arguments are zero, it means the client should decide its own window dimension. This may happen when the compositor needs to configure the state of the surface but doesn't have any information about any previous or expected dimension.

The states listed in the event specify how the width/height arguments should be interpreted, and possibly how it should be drawn.

Clients must send an ack_configure in response to this event. See xdg_surface.configure and xdg_surface.ack_↩ configure for details.

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/xdg-shell-client-protocol.h

16.156 xdg_wm_base_listener Struct Reference

Public Attributes

- void(∗ ping)(void ∗data, struct xdg_wm_base ∗xdg_wm_base, uint32_t serial)

16.156.1 Member Data Documentation

16.156.1.1 ping

```
void(* xdg_wm_base_listener::ping) (void *data, struct xdg_wm_base *xdg_wm_base, uint32_↩
t serial)
```

check if the client is alive

The ping event asks the client if it's still alive. Pass the serial specified in the event back to the compositor by sending a "pong" request back with the specified serial. See xdg_wm_base.ping.

Compositors can use this to determine if the client is still alive. It's unspecified what will happen if the client doesn't respond to the ping request, or in what timeframe. Clients should try to respond in a reasonable amount of time.

A compositor is free to ping in any way it wants, but a client must always respond to any xdg_wm_base object it created.

Parameters

serial	pass this to the pong request

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/xdg-shell-client-protocol.h

16.157 zwp_confined_pointer_v1_listener Struct Reference

Public Attributes

- void(∗ confined)(void ∗data, struct zwp_confined_pointer_v1 ∗zwp_confined_pointer_v1)
- void(∗ unconfined)(void ∗data, struct zwp_confined_pointer_v1 ∗zwp_confined_pointer_v1)

16.157.1 Member Data Documentation

16.157.1.1 confined

```
void(* zwp_confined_pointer_v1_listener::confined) (void *data, struct zwp_confined_pointer_v1
*zwp_confined_pointer_v1)
```

pointer confined

Notification that the pointer confinement of the seat's pointer is activated.

16.157.1.2 unconfined

```
void(* zwp_confined_pointer_v1_listener::unconfined) (void *data, struct zwp_confined_pointer↩
_v1 *zwp_confined_pointer_v1)
```

pointer unconfined

Notification that the pointer confinement of the seat's pointer is no longer active. If this is a oneshot pointer confinement (see wp_pointer_constraints.lifetime) this object is now defunct and should be destroyed. If this is a persistent pointer confinement (see wp_pointer_constraints.lifetime) this pointer confinement may again reactivate in the future.

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/pointer-constraints-unstable-v1-client-protocol.h

16.158 zwp_locked_pointer_v1_listener Struct Reference

Public Attributes

- void(∗ locked)(void ∗data, struct zwp_locked_pointer_v1 ∗zwp_locked_pointer_v1)
- void(∗ unlocked)(void ∗data, struct zwp_locked_pointer_v1 ∗zwp_locked_pointer_v1)

16.158.1 Member Data Documentation

16.158.1.1 locked

```
void(* zwp_locked_pointer_v1_listener::locked) (void *data, struct zwp_locked_pointer_v1 *zwp↩
_locked_pointer_v1)
```

lock activation event

Notification that the pointer lock of the seat's pointer is activated.

16.158.1.2 unlocked

```
void(* zwp_locked_pointer_v1_listener::unlocked) (void *data, struct zwp_locked_pointer_v1
*zwp_locked_pointer_v1)
```

lock deactivation event

Notification that the pointer lock of the seat's pointer is no longer active. If this is a oneshot pointer lock (see wp_pointer_constraints.lifetime) this object is now defunct and should be destroyed. If this is a persistent pointer lock (see wp_pointer_constraints.lifetime) this pointer lock may again reactivate in the future.

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/pointer-constraints-unstable-v1-client-protocol.h

16.159 zwp_relative_pointer_v1_listener Struct Reference

Public Attributes

- void(* relative_motion)(void *data, struct zwp_relative_pointer_v1 *zwp_relative_pointer_v1, uint32_t utime_hi, uint32_t utime_lo, wl_fixed_t dx, wl_fixed_t dy, wl_fixed_t dx_unaccel, wl_fixed_t dy_unaccel)

16.159.1 Member Data Documentation

16.159.1.1 relative_motion

```
void(* zwp_relative_pointer_v1_listener::relative_motion) (void *data, struct zwp_relative_
pointer_v1 *zwp_relative_pointer_v1, uint32_t utime_hi, uint32_t utime_lo, wl_fixed_t dx, wl_
fixed_t dy, wl_fixed_t dx_unaccel, wl_fixed_t dy_unaccel)
```

relative pointer motion

Relative x/y pointer motion from the pointer of the seat associated with this object.

A relative motion is in the same dimension as regular wl_pointer motion events, except they do not represent an absolute position. For example, moving a pointer from (x, y) to (x', y') would have the equivalent relative motion (x' - x, y' - y). If a pointer motion caused the absolute pointer position to be clipped by for example the edge of the monitor, the relative motion is unaffected by the clipping and will represent the unclipped motion.

This event also contains non-accelerated motion deltas. The non-accelerated delta is, when applicable, the regular pointer motion delta as it was before having applied motion acceleration and other transformations such as normalization.

Note that the non-accelerated delta does not represent 'raw' events as they were read from some device. Pointer motion acceleration is device- and configuration-specific and non-accelerated deltas and accelerated deltas may have the same value on some devices.

Relative motions are not coupled to wl_pointer.motion events, and can be sent in combination with such events, but also independently. There may also be scenarios where wl_pointer.motion is sent, but there is no relative motion. The order of an absolute and relative motion event originating from the same physical motion is not guaranteed.

If the client needs button events or focus state, it can receive them from a wl_pointer object of the same seat that the wp_relative_pointer object is associated with.

Parameters

utime_hi	high 32 bits of a 64 bit timestamp with microsecond granularity
utime_lo	low 32 bits of a 64 bit timestamp with microsecond granularity
dx	the x component of the motion vector
dy	the y component of the motion vector
dx_unaccel	the x component of the unaccelerated motion vector
dy_unaccel	the y component of the unaccelerated motion vector

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/relative-pointer-unstable-v1-client-protocol.h

16.160 zxdg_popup_v6_listener Struct Reference

Public Attributes

- void(∗ configure)(void ∗data, struct zxdg_popup_v6 ∗zxdg_popup_v6, int32_t x, int32_t y, int32_t width, int32_t height)
- void(∗ popup_done)(void ∗data, struct zxdg_popup_v6 ∗zxdg_popup_v6)

16.160.1 Member Data Documentation

16.160.1.1 configure

```
void(* zxdg_popup_v6_listener::configure) (void *data, struct zxdg_popup_v6 *zxdg_popup_v6,
int32_t x, int32_t y, int32_t width, int32_t height)
```

configure the popup surface

This event asks the popup surface to configure itself given the configuration. The configured state should not be applied immediately. See xdg_surface.configure for details.

The x and y arguments represent the position the popup was placed at given the xdg_positioner rule, relative to the upper left corner of the window geometry of the parent surface.

Parameters

x	x position relative to parent surface window geometry
y	y position relative to parent surface window geometry
width	window geometry width
height	window geometry height

16.160.1.2 popup_done

```
void(* zxdg_popup_v6_listener::popup_done) (void *data, struct zxdg_popup_v6 *zxdg_popup_v6)
```

popup interaction is done

The popup_done event is sent out when a popup is dismissed by the compositor. The client should destroy the xdg_popup object at this point.

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/xdg-shell-unstable-v6-client-protocol.h

16.161 zxdg_shell_v6_listener Struct Reference

Public Attributes

- void(∗ ping)(void ∗data, struct zxdg_shell_v6 ∗zxdg_shell_v6, uint32_t serial)

16.161.1 Member Data Documentation

16.161.1.1 ping

```
void(* zxdg_shell_v6_listener::ping) (void *data, struct zxdg_shell_v6 *zxdg_shell_v6, uint32↩
_t serial)
```

check if the client is alive

The ping event asks the client if it's still alive. Pass the serial specified in the event back to the compositor by sending a "pong" request back with the specified serial. See xdg_shell.ping.

Compositors can use this to determine if the client is still alive. It's unspecified what will happen if the client doesn't respond to the ping request, or in what timeframe. Clients should try to respond in a reasonable amount of time.

A compositor is free to ping in any way it wants, but a client must always respond to any xdg_shell object it created.

Parameters

serial	pass this to the pong request

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/xdg-shell-unstable-v6-client-protocol.h

16.162 zxdg_surface_v6_listener Struct Reference

Public Attributes

- void(∗ configure)(void ∗data, struct zxdg_surface_v6 ∗zxdg_surface_v6, uint32_t serial)

16.162.1 Member Data Documentation

16.162.1.1 configure

```
void(* zxdg_surface_v6_listener::configure) (void *data, struct zxdg_surface_v6 *zxdg_surface↩
_v6, uint32_t serial)
```

suggest a surface change

The configure event marks the end of a configure sequence. A configure sequence is a set of one or more events configuring the state of the xdg_surface, including the final xdg_surface.configure event.

Where applicable, xdg_surface surface roles will during a configure sequence extend this event as a latched state sent as events before the xdg_surface.configure event. Such events should be considered to make up a set of atomically applied configuration states, where the xdg_surface.configure commits the accumulated state.

Clients should arrange their surface for the new states, and then send an ack_configure request with the serial sent in this configure event at some point before committing the new surface.

If the client receives multiple configure events before it can respond to one, it is free to discard all but the last event it received.

Parameters

serial	serial of the configure event

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/xdg-shell-unstable-v6-client-protocol.h

16.163 zxdg_toplevel_decoration_v1_listener Struct Reference

Public Attributes

- void(∗ configure)(void ∗data, struct zxdg_toplevel_decoration_v1 ∗zxdg_toplevel_decoration_v1, uint32_↩ t mode)

16.163.1 Member Data Documentation

16.163.1.1 configure

```
void(* zxdg_toplevel_decoration_v1_listener::configure) (void *data, struct zxdg_toplevel_↩
decoration_v1 *zxdg_toplevel_decoration_v1, uint32_t mode)
```

suggest a surface change

The configure event asks the client to change its decoration mode. The configured state should not be applied immediately. Clients must send an ack_configure in response to this event. See xdg_surface.configure and xdg_↩ surface.ack_configure for details.

A configure event can be sent at any time. The specified mode must be obeyed by the client.

Parameters

mode	the decoration mode

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/xdg-decoration-unstable-v1-client-protocol.h

16.164 zxdg_toplevel_v6_listener Struct Reference

Public Attributes

- void(∗ configure)(void ∗data, struct zxdg_toplevel_v6 ∗zxdg_toplevel_v6, int32_t width, int32_t height, struct wl_array ∗states)
- void(∗ close)(void ∗data, struct zxdg_toplevel_v6 ∗zxdg_toplevel_v6)

16.164.1 Member Data Documentation

16.164.1.1 close

```
void(* zxdg_toplevel_v6_listener::close) (void *data, struct zxdg_toplevel_v6 *zxdg_toplevel_↩
v6)
```

surface wants to be closed

The close event is sent by the compositor when the user wants the surface to be closed. This should be equivalent to the user clicking the close button in client-side decorations, if your application has any.

This is only a request that the user intends to close the window. The client may choose to ignore this request, or show a dialog to ask the user to save their data, etc.

16.164.1.2 configure

```
void(* zxdg_toplevel_v6_listener::configure) (void *data, struct zxdg_toplevel_v6 *zxdg_↩
toplevel_v6, int32_t width, int32_t height, struct wl_array *states)
```

suggest a surface change

This configure event asks the client to resize its toplevel surface or to change its state. The configured state should not be applied immediately. See xdg_surface.configure for details.

The width and height arguments specify a hint to the window about how its surface should be resized in window geometry coordinates. See set_window_geometry.

If the width or height arguments are zero, it means the client should decide its own window dimension. This may happen when the compositor needs to configure the state of the surface but doesn't have any information about any previous or expected dimension.

The states listed in the event specify how the width/height arguments should be interpreted, and possibly how it should be drawn.

Clients must send an ack_configure in response to this event. See xdg_surface.configure and xdg_surface.ack_↩ configure for details.

The documentation for this struct was generated from the following file:

- cmake-build-debug/build/sdl/wayland-generated-protocols/xdg-shell-unstable-v6-client-protocol.h

Chapter 17

File Documentation

17.1 Resources/XML/Generated/common.hxx File Reference

Generated from common.xsd.

```
#include <xsd/cxx/config.hxx>
#include <xsd/cxx/pre.hxx>
#include <xsd/cxx/xml/char-utf8.hxx>
#include <xsd/cxx/tree/exceptions.hxx>
#include <xsd/cxx/tree/elements.hxx>
#include <xsd/cxx/tree/types.hxx>
#include <xsd/cxx/xml/error-handler.hxx>
#include <xsd/cxx/xml/dom/auto-ptr.hxx>
#include <xsd/cxx/tree/parsing.hxx>
#include <xsd/cxx/tree/parsing/byte.hxx>
#include <xsd/cxx/tree/parsing/unsigned-byte.hxx>
#include <xsd/cxx/tree/parsing/short.hxx>
#include <xsd/cxx/tree/parsing/unsigned-short.hxx>
#include <xsd/cxx/tree/parsing/int.hxx>
#include <xsd/cxx/tree/parsing/unsigned-int.hxx>
#include <xsd/cxx/tree/parsing/long.hxx>
#include <xsd/cxx/tree/parsing/unsigned-long.hxx>
#include <xsd/cxx/tree/parsing/boolean.hxx>
#include <xsd/cxx/tree/parsing/float.hxx>
#include <xsd/cxx/tree/parsing/double.hxx>
#include <xsd/cxx/tree/parsing/decimal.hxx>
#include <memory>
#include <limits>
#include <algorithm>
#include <utility>
#include <xsd/cxx/tree/containers.hxx>
#include <xsd/cxx/tree/list.hxx>
#include <xsd/cxx/xml/dom/parsing-header.hxx>
#include <iosfwd>
#include <xercesc/sax/InputSource.hpp>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMErrorHandler.hpp>
#include <xsd/cxx/post.hxx>
```

Classes

- class Common::baseResources

  Class corresponding to the baseResources schema type.
- class Common::alpha

  Class corresponding to the alpha schema type.
- class Common::preloadResources

  Class corresponding to the preloadResources schema type.
- class Common::resources

  Class corresponding to the resources schema type.
- class Common::events

  Class corresponding to the events schema type.
- class Common::position

  Class corresponding to the position schema type.
- class Common::size

  Class corresponding to the size schema type.
- class Common::color

  Class corresponding to the color schema type.
- class Common::font

  Class corresponding to the font schema type.
- class Common::onEnter

  Class corresponding to the onEnter schema type.
- class Common::onLeave

  Class corresponding to the onLeave schema type.
- class Common::onAttacked

  Class corresponding to the onAttacked schema type.
- class Common::onDestroyed

  Class corresponding to the onDestroyed schema type.
- class Common::onAttack

  Class corresponding to the onAttack schema type.
- class Common::onClick

  Class corresponding to the onClick schema type.

Namespaces

- xml_schema

  C++ namespace for the http://www.w3.org/2001/XMLSchema schema namespace.
- xml_schema::dom

  DOM interaction.
- Common

  C++ namespace for the Common schema namespace.

Typedefs

- typedef ::xsd::cxx::tree::type xml_schema::type

  C++ type corresponding to the anyType XML Schema built-in type.
- typedef ::xsd::cxx::tree::simple_type< char, type > xml_schema::simple_type

  C++ type corresponding to the anySimpleType XML Schema built-in type.
- typedef ::xsd::cxx::tree::type xml_schema::container

  Alias for the anyType type.
- typedef signed char xml_schema::byte

  C++ type corresponding to the byte XML Schema built-in type.
- typedef unsigned char xml_schema::unsigned_byte

  C++ type corresponding to the unsignedByte XML Schema built-in type.
- typedef short xml_schema::short_

  C++ type corresponding to the short XML Schema built-in type.
- typedef unsigned short xml_schema::unsigned_short

  C++ type corresponding to the unsignedShort XML Schema built-in type.
- typedef int xml_schema::int_

  C++ type corresponding to the int XML Schema built-in type.
- typedef unsigned int xml_schema::unsigned_int

  C++ type corresponding to the unsignedInt XML Schema built-in type.
- typedef long long xml_schema::long_

  C++ type corresponding to the long XML Schema built-in type.
- typedef unsigned long long xml_schema::unsigned_long

  C++ type corresponding to the unsignedLong XML Schema built-in type.
- typedef long long xml_schema::integer

  C++ type corresponding to the integer XML Schema built-in type.
- typedef long long xml_schema::non_positive_integer

  C++ type corresponding to the nonPositiveInteger XML Schema built-in type.
- typedef unsigned long long xml_schema::non_negative_integer

  C++ type corresponding to the nonNegativeInteger XML Schema built-in type.
- typedef unsigned long long xml_schema::positive_integer

  C++ type corresponding to the positiveInteger XML Schema built-in type.
- typedef long long xml_schema::negative_integer

  C++ type corresponding to the negativeInteger XML Schema built-in type.
- typedef bool xml_schema::boolean

  C++ type corresponding to the boolean XML Schema built-in type.
- typedef float xml_schema::float_

  C++ type corresponding to the float XML Schema built-in type.
- typedef double xml_schema::double_

  C++ type corresponding to the double XML Schema built-in type.
- typedef double xml_schema::decimal

  C++ type corresponding to the decimal XML Schema built-in type.
- typedef ::xsd::cxx::tree::string< char, simple_type > xml_schema::string

  C++ type corresponding to the string XML Schema built-in type.
- typedef ::xsd::cxx::tree::normalized_string< char, string > xml_schema::normalized_string

  C++ type corresponding to the normalizedString XML Schema built-in type.
- typedef ::xsd::cxx::tree::token< char, normalized_string > xml_schema::token

  C++ type corresponding to the token XML Schema built-in type.
- typedef ::xsd::cxx::tree::name< char, token > xml_schema::name

  C++ type corresponding to the Name XML Schema built-in type.
- typedef ::xsd::cxx::tree::nmtoken< char, token > xml_schema::nmtoken

C++ type corresponding to the NMTOKEN XML Schema built-in type.

- typedef ::xsd::cxx::tree::nmtokens< char, simple_type, nmtoken > xml_schema::nmtokens

  C++ type corresponding to the NMTOKENS XML Schema built-in type.
- typedef ::xsd::cxx::tree::ncname< char, name > xml_schema::ncname

  C++ type corresponding to the NCName XML Schema built-in type.
- typedef ::xsd::cxx::tree::language< char, token > xml_schema::language

  C++ type corresponding to the language XML Schema built-in type.
- typedef ::xsd::cxx::tree::id< char, ncname > xml_schema::id

  C++ type corresponding to the ID XML Schema built-in type.
- typedef ::xsd::cxx::tree::idref< char, ncname, type > xml_schema::idref

  C++ type corresponding to the IDREF XML Schema built-in type.
- typedef ::xsd::cxx::tree::idrefs< char, simple_type, idref > xml_schema::idrefs

  C++ type corresponding to the IDREFS XML Schema built-in type.
- typedef ::xsd::cxx::tree::uri< char, simple_type > xml_schema::uri

  C++ type corresponding to the anyURI XML Schema built-in type.
- typedef ::xsd::cxx::tree::qname< char, simple_type, uri, ncname > xml_schema::qname

  C++ type corresponding to the QName XML Schema built-in type.
- typedef ::xsd::cxx::tree::buffer< char > xml_schema::buffer

  Binary buffer type.
- typedef ::xsd::cxx::tree::base64_binary< char, simple_type > xml_schema::base64_binary

  C++ type corresponding to the base64Binary XML Schema built-in type.
- typedef ::xsd::cxx::tree::hex_binary< char, simple_type > xml_schema::hex_binary

  C++ type corresponding to the hexBinary XML Schema built-in type.
- typedef ::xsd::cxx::tree::time_zone xml_schema::time_zone

  Time zone type.
- typedef ::xsd::cxx::tree::date< char, simple_type > xml_schema::date

  C++ type corresponding to the date XML Schema built-in type.
- typedef ::xsd::cxx::tree::date_time< char, simple_type > xml_schema::date_time

  C++ type corresponding to the dateTime XML Schema built-in type.
- typedef ::xsd::cxx::tree::duration< char, simple_type > xml_schema::duration

  C++ type corresponding to the duration XML Schema built-in type.
- typedef ::xsd::cxx::tree::gday< char, simple_type > xml_schema::gday

  C++ type corresponding to the gDay XML Schema built-in type.
- typedef ::xsd::cxx::tree::gmonth< char, simple_type > xml_schema::gmonth

  C++ type corresponding to the gMonth XML Schema built-in type.
- typedef ::xsd::cxx::tree::gmonth_day< char, simple_type > xml_schema::gmonth_day

  C++ type corresponding to the gMonthDay XML Schema built-in type.
- typedef ::xsd::cxx::tree::gyear< char, simple_type > xml_schema::gyear

  C++ type corresponding to the gYear XML Schema built-in type.
- typedef ::xsd::cxx::tree::gyear_month< char, simple_type > xml_schema::gyear_month

  C++ type corresponding to the gYearMonth XML Schema built-in type.
- typedef ::xsd::cxx::tree::time< char, simple_type > xml_schema::time

  C++ type corresponding to the time XML Schema built-in type.
- typedef ::xsd::cxx::tree::entity< char, ncname > xml_schema::entity

  C++ type corresponding to the ENTITY XML Schema built-in type.
- typedef ::xsd::cxx::tree::entities< char, simple_type, entity > xml_schema::entities

  C++ type corresponding to the ENTITIES XML Schema built-in type.
- typedef ::xsd::cxx::tree::content_order xml_schema::content_order

  Content order sequence entry.
- typedef ::xsd::cxx::tree::flags xml_schema::flags

  Parsing and serialization flags.

- typedef ::xsd::cxx::tree::properties< char > xml_schema::properties

  Parsing properties.
- typedef ::xsd::cxx::tree::severity xml_schema::severity

  Error severity.
- typedef ::xsd::cxx::tree::error< char > xml_schema::error

  Error condition.
- typedef ::xsd::cxx::tree::diagnostics< char > xml_schema::diagnostics

  List of error conditions.
- typedef ::xsd::cxx::tree::exception< char > xml_schema::exception

  Root of the C++/Tree exception hierarchy.
- typedef ::xsd::cxx::tree::bounds< char > xml_schema::bounds

  Exception indicating that the size argument exceeds the capacity argument.
- typedef ::xsd::cxx::tree::duplicate_id< char > xml_schema::duplicate_id

  Exception indicating that a duplicate ID value was encountered in the object model.
- typedef ::xsd::cxx::tree::parsing< char > xml_schema::parsing

  Exception indicating a parsing failure.
- typedef ::xsd::cxx::tree::expected_element< char > xml_schema::expected_element

  Exception indicating that an expected element was not encountered.
- typedef ::xsd::cxx::tree::unexpected_element< char > xml_schema::unexpected_element

  Exception indicating that an unexpected element was encountered.
- typedef ::xsd::cxx::tree::expected_attribute< char > xml_schema::expected_attribute

  Exception indicating that an expected attribute was not encountered.
- typedef ::xsd::cxx::tree::unexpected_enumerator< char > xml_schema::unexpected_enumerator

  Exception indicating that an unexpected enumerator was encountered.
- typedef ::xsd::cxx::tree::expected_text_content< char > xml_schema::expected_text_content

  Exception indicating that the text content was expected for an element.
- typedef ::xsd::cxx::tree::no_prefix_mapping< char > xml_schema::no_prefix_mapping

  Exception indicating that a prefix-namespace mapping was not provided.
- typedef ::xsd::cxx::xml::error_handler< char > xml_schema::error_handler

  Error handler callback interface.

Functions

Parsing functions for the preloadResources document root.

- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (::std::istream &is, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with an error handler.

- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::preloadResources > Common::preloadResources_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the resources document root.

- ::std::unique_ptr< ::Common::resources > Common::resources_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.

- ::std::unique_ptr< ::Common::resources > Common::resources_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (::xercesc::InputSource &is, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (::xercesc::InputSource &is, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (::xercesc::InputSource &is, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (const ::xercesc::DOMDocument &d, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::resources > Common::resources_ (::xml_schema::dom::unique_↩ ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the events document root.

- ::std::unique_ptr< ::Common::events > Common::events_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::events > Common::events_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::events > Common::events_ (const ::std::string &uri, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::events > Common::events_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::events > Common::events_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::events > Common::events_ (::std::istream &is, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::events > Common::events_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::events > Common::events_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::events > Common::events_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.

- ::std::unique_ptr< ::Common::events > Common::events_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::events > Common::events_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::events > Common::events_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::events > Common::events_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::events > Common::events_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the position document root.

- ::std::unique_ptr< ::Common::position > Common::position_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::position > Common::position_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::position > Common::position_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::position > Common::position_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::position > Common::position_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::position > Common::position_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::position > Common::position_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::position > Common::position_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::position > Common::position_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::position > Common::position_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::position > Common::position_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::position > Common::position_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.

- ::std::unique_ptr< ::Common::position > Common::position_ (const ::xercesc::DOMDocument &d, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::position > Common::position_ (::xml_schema::dom::unique_ptr< ↩ ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the size document root.

- ::std::unique_ptr< ::Common::size > Common::size_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::size > Common::size_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::size > Common::size_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::size > Common::size_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::size > Common::size_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::size > Common::size_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::size > Common::size_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::size > Common::size_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::size > Common::size_ (::std::istream &is, const ::std::string &id, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::size > Common::size_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::size > Common::size_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::size > Common::size_ (::xercesc::InputSource &is, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::size > Common::size_ (const ::xercesc::DOMDocument &d, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::size > Common::size_ (::xml_schema::dom::unique_ptr< ::xercesc::DO↩ MDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the color document root.

- ::std::unique_ptr< ::Common::color > Common::color_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::color > Common::color_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::color > Common::color_ (const ::std::string &uri, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::color > Common::color_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Common::color > Common::color_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::color > Common::color_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::color > Common::color_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::color > Common::color_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::color > Common::color_ (::std::istream &is, const ::std::string &id, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::color > Common::color_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::color > Common::color_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::color > Common::color_ (::xercesc::InputSource &is, ::xercesc::DOM↩ ErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::color > Common::color_ (const ::xercesc::DOMDocument &d, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::color > Common::color_ (::xml_schema::dom::unique_ptr< ::xercesc::D↩ OMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the font document root.

- ::std::unique_ptr< ::Common::font > Common::font_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Common::font > Common::font_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Common::font > Common::font_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::font > Common::font_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

Parse a standard input stream.

- ::std::unique_ptr< ::Common::font > Common::font_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Common::font > Common::font_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::font > Common::font_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Common::font > Common::font_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Common::font > Common::font_ (::std::istream &is, const ::std::string &id, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::font > Common::font_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Common::font > Common::font_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Common::font > Common::font_ (::xercesc::InputSource &is, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Common::font > Common::font_ (const ::xercesc::DOMDocument &d, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Common::font > Common::font_ (::xml_schema::dom::unique_ptr< ::xercesc::DO↩ MDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Variables

- const XMLCh ∗const xml_schema::dom::tree_node_key = ::xsd::cxx::tree::user_data_keys::node

  DOM user data key for back pointers to tree nodes.

17.1.1 Detailed Description

Generated from common.xsd.

17.2 Resources/XML/Generated/components.hxx File Reference

Generated from components.xsd.

```
#include <xsd/cxx/config.hxx>
#include <xsd/cxx/pre.hxx>
#include <xsd/cxx/xml/char-utf8.hxx>
#include <xsd/cxx/tree/exceptions.hxx>
#include <xsd/cxx/tree/elements.hxx>
```

```
#include <xsd/cxx/tree/types.hxx>
#include <xsd/cxx/xml/error-handler.hxx>
#include <xsd/cxx/xml/dom/auto-ptr.hxx>
#include <xsd/cxx/tree/parsing.hxx>
#include <xsd/cxx/tree/parsing/byte.hxx>
#include <xsd/cxx/tree/parsing/unsigned-byte.hxx>
#include <xsd/cxx/tree/parsing/short.hxx>
#include <xsd/cxx/tree/parsing/unsigned-short.hxx>
#include <xsd/cxx/tree/parsing/int.hxx>
#include <xsd/cxx/tree/parsing/unsigned-int.hxx>
#include <xsd/cxx/tree/parsing/long.hxx>
#include <xsd/cxx/tree/parsing/unsigned-long.hxx>
#include <xsd/cxx/tree/parsing/boolean.hxx>
#include <xsd/cxx/tree/parsing/float.hxx>
#include <xsd/cxx/tree/parsing/double.hxx>
#include <xsd/cxx/tree/parsing/decimal.hxx>
#include <memory>
#include <limits>
#include <algorithm>
#include <utility>
#include <xsd/cxx/tree/containers.hxx>
#include <xsd/cxx/tree/list.hxx>
#include <xsd/cxx/xml/dom/parsing-header.hxx>
#include "common.hxx"
#include <iosfwd>
#include <xercesc/sax/InputSource.hpp>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMErrorHandler.hpp>
#include <xsd/cxx/post.hxx>
```

Classes

- class Components::componentName

  Class corresponding to the componentName schema type.
- class Components::component

  Class corresponding to the component schema type.
- class Components::floatCap

  Class corresponding to the floatCap schema type.
- class Components::bodyType

  Class corresponding to the bodyType schema type.
- class Components::bodyShape

  Class corresponding to the bodyShape schema type.
- class Components::transformComponent

  Class corresponding to the transformComponent schema type.
- class Components::renderComponent

  Class corresponding to the renderComponent schema type.
- class Components::physicsComponent

  Class corresponding to the physicsComponent schema type.
- class Components::characterComponent

  Class corresponding to the characterComponent schema type.
- class Components::explosionCrate

  Class corresponding to the explosionCrate schema type.
- class Components::bulletComponent

Class corresponding to the bulletComponent schema type.

- class Components::nextLevelComponent

  Class corresponding to the nextLevelComponent schema type.

- class Components::circle

  Class corresponding to the circle schema type.

- class Components::box

  Class corresponding to the box schema type.

Namespaces

- xml_schema

  C++ namespace for the http://www.w3.org/2001/XMLSchema schema namespace.

- xml_schema::dom

  DOM interaction.

- Components

  C++ namespace for the Components schema namespace.

17.2.1 Detailed Description

Generated from components.xsd.

17.3 Resources/XML/Generated/level-resources.hxx File Reference

Generated from level-resources.xsd.

```
#include <xsd/cxx/config.hxx>
#include <xsd/cxx/pre.hxx>
#include <xsd/cxx/xml/char-utf8.hxx>
#include <xsd/cxx/tree/exceptions.hxx>
#include <xsd/cxx/tree/elements.hxx>
#include <xsd/cxx/tree/types.hxx>
#include <xsd/cxx/xml/error-handler.hxx>
#include <xsd/cxx/xml/dom/auto-ptr.hxx>
#include <xsd/cxx/tree/parsing.hxx>
#include <xsd/cxx/tree/parsing/byte.hxx>
#include <xsd/cxx/tree/parsing/unsigned-byte.hxx>
#include <xsd/cxx/tree/parsing/short.hxx>
#include <xsd/cxx/tree/parsing/unsigned-short.hxx>
#include <xsd/cxx/tree/parsing/int.hxx>
#include <xsd/cxx/tree/parsing/unsigned-int.hxx>
#include <xsd/cxx/tree/parsing/long.hxx>
#include <xsd/cxx/tree/parsing/unsigned-long.hxx>
#include <xsd/cxx/tree/parsing/boolean.hxx>
#include <xsd/cxx/tree/parsing/float.hxx>
#include <xsd/cxx/tree/parsing/double.hxx>
#include <xsd/cxx/tree/parsing/decimal.hxx>
#include <memory>
#include <limits>
#include <algorithm>
#include <utility>
#include <xsd/cxx/tree/containers.hxx>
```

```
#include <xsd/cxx/tree/list.hxx>
#include <xsd/cxx/xml/dom/parsing-header.hxx>
#include "common.hxx"
#include "objects.hxx"
#include <iosfwd>
#include <xercesc/sax/InputSource.hpp>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMErrorHandler.hpp>
#include <xsd/cxx/post.hxx>
```

Classes

- class LevelResources::level

  Class corresponding to the level schema type.

Namespaces

- xml_schema

  C++ namespace for the http://www.w3.org/2001/XMLSchema schema namespace.
- xml_schema::dom

  DOM interaction.
- LevelResources

  C++ namespace for the LevelResources schema namespace.

Functions

Parsing functions for the level document root.

- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.

- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (::xercesc::InputSource &is, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (::xercesc::InputSource &is, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (::xercesc::InputSource &is, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::LevelResources::level > LevelResources::level_ (::xml_schema::dom::unique_↩ ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

17.3.1 Detailed Description

Generated from level-resources.xsd.

17.4 Resources/XML/Generated/menu.hxx File Reference

Generated from menu.xsd.

```
#include <xsd/cxx/config.hxx>
#include <xsd/cxx/pre.hxx>
#include <xsd/cxx/xml/char-utf8.hxx>
#include <xsd/cxx/tree/exceptions.hxx>
#include <xsd/cxx/tree/elements.hxx>
#include <xsd/cxx/tree/types.hxx>
#include <xsd/cxx/xml/error-handler.hxx>
#include <xsd/cxx/xml/dom/auto-ptr.hxx>
#include <xsd/cxx/tree/parsing.hxx>
#include <xsd/cxx/tree/parsing/byte.hxx>
#include <xsd/cxx/tree/parsing/unsigned-byte.hxx>
#include <xsd/cxx/tree/parsing/short.hxx>
#include <xsd/cxx/tree/parsing/unsigned-short.hxx>
#include <xsd/cxx/tree/parsing/int.hxx>
#include <xsd/cxx/tree/parsing/unsigned-int.hxx>
#include <xsd/cxx/tree/parsing/long.hxx>
#include <xsd/cxx/tree/parsing/unsigned-long.hxx>
#include <xsd/cxx/tree/parsing/boolean.hxx>
#include <xsd/cxx/tree/parsing/float.hxx>
```

```
#include <xsd/cxx/tree/parsing/double.hxx>
#include <xsd/cxx/tree/parsing/decimal.hxx>
#include <memory>
#include <limits>
#include <algorithm>
#include <utility>
#include <xsd/cxx/tree/containers.hxx>
#include <xsd/cxx/tree/list.hxx>
#include <xsd/cxx/xml/dom/parsing-header.hxx>
#include "common.hxx"
#include <iosfwd>
#include <xercesc/sax/InputSource.hpp>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMErrorHandler.hpp>
#include <xsd/cxx/post.hxx>
```

Classes

- class Menu::menu

  Class corresponding to the menu schema type.
- class Menu::buttons

  Class corresponding to the buttons schema type.
- class Menu::texts

  Class corresponding to the texts schema type.
- class Menu::images

  Class corresponding to the images schema type.
- class Menu::boxes

  Class corresponding to the boxes schema type.
- class Menu::button

  Class corresponding to the button schema type.
- class Menu::text

  Class corresponding to the text schema type.
- class Menu::image

  Class corresponding to the image schema type.
- class Menu::box

  Class corresponding to the box schema type.
- class Menu::text1

  Class corresponding to the text1 schema type.
- class Menu::resources

  Class corresponding to the resources schema type.

Namespaces

- xml_schema

  C++ namespace for the http://www.w3.org/2001/XMLSchema schema namespace.
- xml_schema::dom

  DOM interaction.
- Menu

  C++ namespace for the Menu schema namespace.

Functions

Parsing functions for the menu document root.

- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (::std::istream &is, const ::std::string &id, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (::xercesc::InputSource &is, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Menu::menu > Menu::menu_ (::xml_schema::dom::unique_ptr< ::xercesc::DOM↩ Document > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

17.4.1 Detailed Description

Generated from menu.xsd.

17.5 Resources/XML/Generated/objects.hxx File Reference

Generated from objects.xsd.

```
#include <xsd/cxx/config.hxx>
#include <xsd/cxx/pre.hxx>
#include <xsd/cxx/xml/char-utf8.hxx>
#include <xsd/cxx/tree/exceptions.hxx>
#include <xsd/cxx/tree/elements.hxx>
#include <xsd/cxx/tree/types.hxx>
#include <xsd/cxx/xml/error-handler.hxx>
#include <xsd/cxx/xml/dom/auto-ptr.hxx>
#include <xsd/cxx/tree/parsing.hxx>
#include <xsd/cxx/tree/parsing/byte.hxx>
#include <xsd/cxx/tree/parsing/unsigned-byte.hxx>
#include <xsd/cxx/tree/parsing/short.hxx>
#include <xsd/cxx/tree/parsing/unsigned-short.hxx>
#include <xsd/cxx/tree/parsing/int.hxx>
#include <xsd/cxx/tree/parsing/unsigned-int.hxx>
#include <xsd/cxx/tree/parsing/long.hxx>
#include <xsd/cxx/tree/parsing/unsigned-long.hxx>
#include <xsd/cxx/tree/parsing/boolean.hxx>
#include <xsd/cxx/tree/parsing/float.hxx>
#include <xsd/cxx/tree/parsing/double.hxx>
#include <xsd/cxx/tree/parsing/decimal.hxx>
#include <memory>
#include <limits>
#include <algorithm>
#include <utility>
#include <xsd/cxx/tree/containers.hxx>
#include <xsd/cxx/tree/list.hxx>
#include <xsd/cxx/xml/dom/parsing-header.hxx>
#include "components.hxx"
#include <iosfwd>
#include <xercesc/sax/InputSource.hpp>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMErrorHandler.hpp>
#include <xsd/cxx/post.hxx>
```

Classes

- class Objects::object

  Class corresponding to the object schema type.
- class Objects::objectList

  Class corresponding to the objectList schema type.
- class Objects::components

  Class corresponding to the components schema type.

Namespaces

- xml_schema

  C++ namespace for the http://www.w3.org/2001/XMLSchema schema namespace.
- xml_schema::dom

  DOM interaction.
- Objects

  C++ namespace for the Objects schema namespace.

Functions

Parsing functions for the objectList document root.

- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Objects::objectList > Objects::objectList_ (::xml_schema::dom::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

17.5.1 Detailed Description

Generated from objects.xsd.

17.6 Resources/XML/Generated/resources.hxx File Reference

Generated from resources.xsd.

```
#include <xsd/cxx/config.hxx>
#include <xsd/cxx/pre.hxx>
#include <xsd/cxx/xml/char-utf8.hxx>
#include <xsd/cxx/tree/exceptions.hxx>
#include <xsd/cxx/tree/elements.hxx>
#include <xsd/cxx/tree/types.hxx>
#include <xsd/cxx/xml/error-handler.hxx>
#include <xsd/cxx/xml/dom/auto-ptr.hxx>
#include <xsd/cxx/tree/parsing.hxx>
#include <xsd/cxx/tree/parsing/byte.hxx>
#include <xsd/cxx/tree/parsing/unsigned-byte.hxx>
#include <xsd/cxx/tree/parsing/short.hxx>
#include <xsd/cxx/tree/parsing/unsigned-short.hxx>
#include <xsd/cxx/tree/parsing/int.hxx>
#include <xsd/cxx/tree/parsing/unsigned-int.hxx>
#include <xsd/cxx/tree/parsing/long.hxx>
#include <xsd/cxx/tree/parsing/unsigned-long.hxx>
#include <xsd/cxx/tree/parsing/boolean.hxx>
#include <xsd/cxx/tree/parsing/float.hxx>
#include <xsd/cxx/tree/parsing/double.hxx>
#include <xsd/cxx/tree/parsing/decimal.hxx>
#include <memory>
#include <limits>
#include <algorithm>
#include <utility>
#include <xsd/cxx/tree/containers.hxx>
#include <xsd/cxx/tree/list.hxx>
#include <xsd/cxx/xml/dom/parsing-header.hxx>
#include "common.hxx"
#include "components.hxx"
#include <iosfwd>
#include <xercesc/sax/InputSource.hpp>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMErrorHandler.hpp>
#include <xsd/cxx/post.hxx>
```

Classes

- class GameResources::baseGameResource

  Class corresponding to the baseGameResource schema type.
- class GameResources::resources

  Class corresponding to the resources schema type.
- class GameResources::textures

  Class corresponding to the textures schema type.
- class GameResources::sprites

  Class corresponding to the sprites schema type.
- class GameResources::sounds

  Class corresponding to the sounds schema type.
- class GameResources::music

Class corresponding to the music schema type.

- class GameResources::scenes

  Class corresponding to the scenes schema type.
- class GameResources::levels

  Class corresponding to the levels schema type.
- class GameResources::objectLists

  Class corresponding to the objectLists schema type.
- class GameResources::texture

  Class corresponding to the texture schema type.
- class GameResources::sprite

  Class corresponding to the sprite schema type.
- class GameResources::sound

  Class corresponding to the sound schema type.
- class GameResources::music1

  Class corresponding to the music1 schema type.
- class GameResources::scene

  Class corresponding to the scene schema type.
- class GameResources::level

  Class corresponding to the level schema type.
- class GameResources::objectList

  Class corresponding to the objectList schema type.
- class GameResources::pool

  Class corresponding to the pool schema type.

Namespaces

- xml_schema

  C++ namespace for the http://www.w3.org/2001/XMLSchema schema namespace.
- xml_schema::dom

  DOM interaction.
- GameResources

  C++ namespace for the GameResources schema namespace.

Functions

Parsing functions for the resources document root.

- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (::std::istream &is, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.

- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (::std::istream &is, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (::std::istream &is, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (::std::istream &is, const ::std::string &id, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (const ::xercesc::DO↩ MDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::GameResources::resources > GameResources::resources_ (::xml_schema::dom↩ ::unique_ptr< ::xercesc::DOMDocument > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

17.6.1 Detailed Description

Generated from resources.xsd.

17.7 Resources/XML/Generated/sentry.hxx File Reference

Generated from sentry.xsd.

```
#include <xsd/cxx/config.hxx>
#include <xsd/cxx/pre.hxx>
#include <xsd/cxx/xml/char-utf8.hxx>
#include <xsd/cxx/tree/exceptions.hxx>
#include <xsd/cxx/tree/elements.hxx>
#include <xsd/cxx/tree/types.hxx>
#include <xsd/cxx/xml/error-handler.hxx>
#include <xsd/cxx/xml/dom/auto-ptr.hxx>
```

```
#include <xsd/cxx/tree/parsing.hxx>
#include <xsd/cxx/tree/parsing/byte.hxx>
#include <xsd/cxx/tree/parsing/unsigned-byte.hxx>
#include <xsd/cxx/tree/parsing/short.hxx>
#include <xsd/cxx/tree/parsing/unsigned-short.hxx>
#include <xsd/cxx/tree/parsing/int.hxx>
#include <xsd/cxx/tree/parsing/unsigned-int.hxx>
#include <xsd/cxx/tree/parsing/long.hxx>
#include <xsd/cxx/tree/parsing/unsigned-long.hxx>
#include <xsd/cxx/tree/parsing/boolean.hxx>
#include <xsd/cxx/tree/parsing/float.hxx>
#include <xsd/cxx/tree/parsing/double.hxx>
#include <xsd/cxx/tree/parsing/decimal.hxx>
#include <memory>
#include <limits>
#include <algorithm>
#include <utility>
#include <xsd/cxx/tree/containers.hxx>
#include <xsd/cxx/tree/list.hxx>
#include <xsd/cxx/xml/dom/parsing-header.hxx>
#include "common.hxx"
#include <iosfwd>
#include <xercesc/sax/InputSource.hpp>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMErrorHandler.hpp>
#include <xsd/cxx/post.hxx>
```

Classes

- class Sentry::sentry

  Class corresponding to the sentry schema type.

Namespaces

- xml_schema

  C++ namespace for the http://www.w3.org/2001/XMLSchema schema namespace.
- xml_schema::dom

  DOM interaction.
- Sentry

  C++ namespace for the Sentry schema namespace.

Functions

Parsing functions for the sentry document root.

- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.

- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (::std::istream &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (::std::istream &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (::std::istream &is, const ::std::string &id, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (::xercesc::InputSource &is, ::xercesc::DOMError↩ Handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (const ::xercesc::DOMDocument &d, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Sentry::sentry > Sentry::sentry_ (::xml_schema::dom::unique_ptr< ::xercesc::DOM↩ Document > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

17.7.1 Detailed Description

Generated from sentry.xsd.

17.8 Resources/XML/Generated/wall.hxx File Reference

Generated from wall.xsd.

```
#include <xsd/cxx/config.hxx>
#include <xsd/cxx/pre.hxx>
#include <xsd/cxx/xml/char-utf8.hxx>
#include <xsd/cxx/tree/exceptions.hxx>
#include <xsd/cxx/tree/elements.hxx>
#include <xsd/cxx/tree/types.hxx>
#include <xsd/cxx/xml/error-handler.hxx>
```

```
#include <xsd/cxx/xml/dom/auto-ptr.hxx>
#include <xsd/cxx/tree/parsing.hxx>
#include <xsd/cxx/tree/parsing/byte.hxx>
#include <xsd/cxx/tree/parsing/unsigned-byte.hxx>
#include <xsd/cxx/tree/parsing/short.hxx>
#include <xsd/cxx/tree/parsing/unsigned-short.hxx>
#include <xsd/cxx/tree/parsing/int.hxx>
#include <xsd/cxx/tree/parsing/unsigned-int.hxx>
#include <xsd/cxx/tree/parsing/long.hxx>
#include <xsd/cxx/tree/parsing/unsigned-long.hxx>
#include <xsd/cxx/tree/parsing/boolean.hxx>
#include <xsd/cxx/tree/parsing/float.hxx>
#include <xsd/cxx/tree/parsing/double.hxx>
#include <xsd/cxx/tree/parsing/decimal.hxx>
#include <memory>
#include <limits>
#include <algorithm>
#include <utility>
#include <xsd/cxx/tree/containers.hxx>
#include <xsd/cxx/tree/list.hxx>
#include <xsd/cxx/xml/dom/parsing-header.hxx>
#include "common.hxx"
#include <iosfwd>
#include <xercesc/sax/InputSource.hpp>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMErrorHandler.hpp>
#include <xsd/cxx/post.hxx>
```

Classes

- class Walls::walls

  Class corresponding to the walls schema type.
- class Walls::wall

  Class corresponding to the wall schema type.
- class Walls::pricing

  Class corresponding to the pricing schema type.
- class Walls::powers

  Class corresponding to the powers schema type.
- class Walls::upgrade

  Class corresponding to the upgrade schema type.

Namespaces

- xml_schema

  C++ namespace for the http://www.w3.org/2001/XMLSchema schema namespace.
- xml_schema::dom

  DOM interaction.
- Walls

  C++ namespace for the Walls schema namespace.

Functions

Parsing functions for the walls document root.

- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (const ::std::string &uri, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (::std::istream &is, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (::std::istream &is, const ::std::string &id, ↩ ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (::std::istream &is, const ::std::string &id, ↩ ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=↩ ::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Walls::walls > Walls::walls_ (::xml_schema::dom::unique_ptr< ::xercesc::DOM↩ Document > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

Parsing functions for the wall document root.

- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (const ::std::string &uri, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (const ::std::string &uri, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a URI or a local file with an error handler.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (const ::std::string &uri, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

Parse a URI or a local file with a Xerces-C++ DOM error handler.

- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (::std::istream &is, ::xml_schema::flags f=0, const ↩ ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (::std::istream &is, ::xml_schema::error_handler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with an error handler.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (::std::istream &is, ::xercesc::DOMErrorHandler &eh, ↩ ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (::std::istream &is, const ::std::string &id, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (::std::istream &is, const ::std::string &id, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and an error handler.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (::std::istream &is, const ::std::string &id, ::xercesc::DOM↩ ErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a standard input stream with a resource id and a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (::xercesc::InputSource &is, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (::xercesc::InputSource &is, ::xml_schema::error_handler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with an error handler.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (::xercesc::InputSource &is, ::xercesc::DOMErrorHandler &eh, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ input source with a Xerces-C++ DOM error handler.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (const ::xercesc::DOMDocument &d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.
- ::std::unique_ptr< ::Walls::wall > Walls::wall_ (::xml_schema::dom::unique_ptr< ::xercesc::DOM↩ Document > d, ::xml_schema::flags f=0, const ::xml_schema::properties &p=::xml_schema::properties())

  Parse a Xerces-C++ DOM document.

17.8.1 Detailed Description

Generated from wall.xsd.