

# Package: rotio (via r-universe)

June 19, 2026

**Type** Package

**Title** Pure-R 'OpenTimelineIO' Document Model

**Version** 0.1.0

**Date** 2026-06-12

**Description** A dependency-light 'OpenTimelineIO' (OTIO) <<https://github.com/AcademySoftwareFoundation/OpenTimelineIO>> document layer in pure R. Provides constructors for the OTIO object model (timelines, tracks, clips, gaps, media references, rational times and time ranges), functional builders that return new objects, and readers and writers for canonical '.otio' files through 'jsonlite'. The optional 'RcppOTIO' package validates output against the real 'OpenTimelineIO' C++ library. No compiled code.

**License** Apache License (== 2.0)

**URL** <https://github.com/cornball-ai/rotio>

**BugReports** <https://github.com/cornball-ai/rotio/issues>

**Depends** R (>= 4.4.0)

**Imports** jsonlite

**Additional\_repositories** <https://cornball-ai.github.io/drat>

**Suggests** tinytest, RcppOTIO

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Troy Hernandez [aut, cre] (ORCID: <<https://orcid.org/0009-0005-4248-604X>>), cornball.ai [cph]

**Maintainer** Troy Hernandez <troy@cornball.ai>

**Repository** <https://cran.r-universe.dev>

**Date/Publication** 2026-06-19 12:20:09 UTC

**RemoteUrl** <https://github.com/cran/rotio>

**RemoteRef** HEAD

**RemoteSha** 2aa8aa1f7e4bc301a5bb25d6dd128d10d8cea6c1

## Contents

active_media_reference_key . . . . .	4
add_child . . . . .	5
add_effect . . . . .	5
add_track . . . . .	6
almost_equal . . . . .	7
append_child . . . . .	7
available_range . . . . .	8
children . . . . .	8
clamped . . . . .	9
clear_children . . . . .	10
Clip . . . . .	10
clone . . . . .	11
color . . . . .	11
comment . . . . .	12
contains . . . . .	13
default_media_key . . . . .	13
Effect . . . . .	14
effect_name . . . . .	15
effects . . . . .	15
enabled . . . . .	16
end_frame . . . . .	16
end_time_exclusive . . . . .	17
extended_by . . . . .	18
ExternalReference . . . . .	18
fill . . . . .	19
find_clips . . . . .	20
flatten_stack . . . . .	20
frame_for_time . . . . .	21
from_frames . . . . .	21
from_json_file . . . . .	22
from_json_string . . . . .	22
from_seconds . . . . .	23
from_time_string . . . . .	23
from_timecode . . . . .	24
Gap . . . . .	24
generator_kind . . . . .	25
GeneratorReference . . . . .	26
global_start_time . . . . .	26
has_child . . . . .	27
has_clips . . . . .	28
ImageSequenceReference . . . . .	28
index_of_child . . . . .	29
insert . . . . .	30
insert_child . . . . .	31
intersects . . . . .	31
is_effect . . . . .	32

is_equivalent_to . . . . .	32
is_missing_reference . . . . .	33
is_otio . . . . .	33
is_parent_of . . . . .	34
is_rational_time . . . . .	35
is_unknown_schema . . . . .	35
Item . . . . .	36
kind . . . . .	36
marked_range . . . . .	37
Marker . . . . .	38
media_reference . . . . .	38
media_references . . . . .	39
MediaReference . . . . .	39
metadata . . . . .	40
name . . . . .	41
number_of_images_in_sequence . . . . .	41
overlapping . . . . .	42
overlaps . . . . .	42
overwrite . . . . .	43
parameters . . . . .	44
parent . . . . .	44
presentation_time_for_image_number . . . . .	45
range_from_start_end_time . . . . .	45
range_in_parent . . . . .	46
RationalTime . . . . .	47
read_otiod . . . . .	47
register_downgrade_function . . . . .	48
register_upgrade_function . . . . .	49
remove . . . . .	49
remove_child . . . . .	50
rescaled_to . . . . .	51
ripple . . . . .	51
roll . . . . .	52
schema_name . . . . .	53
schema_version . . . . .	53
SerializableCollection . . . . .	54
set_child . . . . .	54
set_children . . . . .	55
set_media_references . . . . .	56
slice . . . . .	56
slide . . . . .	57
slip . . . . .	58
source_range . . . . .	58
Stack . . . . .	59
start_time . . . . .	60
target_url . . . . .	60
target_url_base . . . . .	61
target_url_for_image_number . . . . .	62

time_scalar . . . . .	62
Timeline . . . . .	63
TimeRange . . . . .	64
TimeTransform . . . . .	64
to_frames . . . . .	65
to_json_file . . . . .	65
to_json_string . . . . .	66
to_seconds . . . . .	67
to_time_string . . . . .	67
to_timecode . . . . .	68
Track . . . . .	68
track_trimmed_to_range . . . . .	69
tracks . . . . .	70
Transition . . . . .	70
transition_type . . . . .	71
trim . . . . .	72
trimmed_range . . . . .	72
trimmed_range_in_parent . . . . .	73
type_version_map . . . . .	74
validate_with_RcppOTIO . . . . .	74
value . . . . .	75
video_tracks . . . . .	75
visible . . . . .	76
visible_range . . . . .	76
write_otiod . . . . .	77

## Index 78

---

active\_media\_reference\_key  
*Active media reference key of a clip*

---

### Description

Active media reference key of a clip

### Usage

```
active_media_reference_key(x)

active_media_reference_key(x) <- value
```

### Arguments

x	A <a href="#">Clip</a> .
value	New active key.

**Value**

The active media reference key, a character string.

**Examples**

```
c1 <- Clip("a", ExternalReference("a.mp4"))
active_media_reference_key(c1)
```

---

add_child	<i>Append a child, functionally (clone + append)</i>
-----------	--

---

**Description**

Returns a new composition: a clone of parent with a clone of child appended. Neither input is mutated (cf. [append\\_child](#), which mutates in place).

**Usage**

```
add_child(parent, child)
```

**Arguments**

parent	A composition (Track/Stack) or collection.
child	An OTIO object.

**Value**

A new composition.

**Examples**

```
v <- add_child(Track("V1"), Clip("a", ExternalReference("a.mp4")))
```

---

add_effect	<i>Append an effect (functional: returns a new object)</i>
------------	--

---

**Description**

Returns a clone of x with effect appended to its effects list; the input is unchanged.

**Usage**

```
add_effect(x, effect)
```

**Arguments**

x                    An item or composition (anything with an effects list).  
 effect              An [Effect](#).

**Value**

A new object of the same class.

**Examples**

```
clip <- Clip("a", ExternalReference("a.mp4"))
clip <- add_effect(clip, LinearTimeWarp(effect_name = "LinearTimeWarp",
                                       time_scalar = 2))
length(effects(clip))
```

---

add_track	<i>Append a track to a timeline, functionally</i>
-----------	---

---

**Description**

Returns a new timeline (clone) with a clone of track appended to its stack. Neither input is mutated.

**Usage**

```
add_track(timeline, track)
```

**Arguments**

timeline            A [Timeline](#).  
 track                A [Track](#).

**Value**

A new timeline.

**Examples**

```
t1 <- add_track(Timeline("demo"), Track("V1", kind = "Video"))
```

---

almost_equal	<i>Are two RationalTimes almost equal?</i>
--------------	--

---

**Description**

Rescales a to b's rate and compares to b's value within delta (in b's rate units), matching opentime.

**Usage**

```
almost_equal(a, b, delta = 0)
```

**Arguments**

delta	Tolerance in b's rate units (default 0).
a, b	RationalTimes.

**Value**

Logical scalar.

**Examples**

```
almost_equal(RationalTime(24, 24), RationalTime(48, 48))
```

---

append_child	<i>Append a child to a composition (in place)</i>
--------------	---

---

**Description**

Mutates x, attaching child and setting its parent. Errors if child already has a parent (use [remove\\_child](#) first, or the functional [add\\_child](#)). Returns x invisibly.

**Usage**

```
append_child(x, child)
```

**Arguments**

x	A composition or collection.
child	An OTIO object with no parent.

**Value**

x, invisibly.

**Examples**

```

trk <- Track("V1")
append_child(trk, Clip("a", ExternalReference("a.mp4")))
length(children(trk))

```

---

available_range	<i>Available range of a media reference or clip</i>
-----------------	---

---

**Description**

For a media reference, the stored available range. For a clip, the active media reference's available range (errors if none is set, matching OTIO).

**Usage**

```

available_range(x)

available_range(x) <- value

```

**Arguments**

x	A media reference or clip.
value	A <a href="#">TimeRange</a> or NULL.

**Value**

A [TimeRange](#) (NULL for a media reference with none set); the setter returns x.

**Examples**

```

ref <- ExternalReference("a.mp4",
  available_range = TimeRange(RationalTime(0, 30), RationalTime(90, 30)))
available_range(ref)

```

---

children	<i>Children of a composition or collection</i>
----------	--

---

**Description**

Children of a composition or collection

**Usage**

```

children(x)

```

**Arguments**

x                    A composition (Track/Stack) or collection.

**Value**

A list of child OTIO objects, in order.

**Examples**

```
trk <- Track("V1")
append_child(trk, Clip("a"))
children(trk)
```

---

clamped

*Clamp a time or range into a bounding TimeRange*

---

**Description**

tr is the bounding range; other is clamped into it (opentime TimeRange::clamped).

**Usage**

```
clamped(tr, other)
```

**Arguments**

tr                    The bounding TimeRange.  
other                 A RationalTime or TimeRange to clamp.

**Value**

A clamped RationalTime or TimeRange, matching other.

**Examples**

```
tr <- TimeRange(RationalTime(0, 24), RationalTime(48, 24))
clamped(tr, RationalTime(96, 24))
```

---

clear_children	<i>Remove all children (in place)</i>
----------------	---------------------------------------

---

**Description**

Remove all children (in place)

**Usage**

```
clear_children(x)
```

**Arguments**

x                    A composition or collection.

**Value**

x, invisibly.

**Examples**

```
trk <- Track("V1")
append_child(trk, Clip("a"))
clear_children(trk)
length(children(trk))
```

---

Clip	<i>Construct a Clip</i>
------	-------------------------

---

**Description**

A media reference plus a source\_range. The reference is stored under the DEFAULT\_MEDIA key (OTIO's multi-reference model).

**Usage**

```
Clip(name, media_reference = MissingReference(), source_range = NULL,
      metadata = NULL)
```

**Arguments**

name                    Clip name.  
media\_reference            A media reference (default MissingReference).  
source\_range            Optional [TimeRange](#).  
metadata                Named list of metadata.

**Value**

A Clip.

**Examples**

```
Clip("a", ExternalReference("a.mp4"),
     source_range = TimeRange(RationalTime(0, 30), RationalTime(90, 30)))
```

---

clone

*Deep-clone an OTIO object*

---

**Description**

Returns a deep copy of `x`. The clone's parent is reset to NULL and its descendants' parent pointers are rewired inside the clone, so it is a fully detached, internally consistent subtree.

**Usage**

```
clone(x)
```

**Arguments**

`x` An OTIO object.

**Value**

A deep copy.

**Examples**

```
t1 <- add_track(Timeline("a"), Track("V1"))
t12 <- clone(t1)
```

---

color

*Get or set an item's display color*

---

**Description**

Get or set an item's display color

**Usage**

```
color(x)
```

```
color(x) <- value
```

**Arguments**

x	An item or composition.
value	A color value (or NULL).

**Value**

The item's color value, or NULL if unset.

**Examples**

```
cl <- Clip("a")
color(cl) <- "RED"
color(cl)
```

---

comment	<i>Comment of a Marker</i>
---------	----------------------------

---

**Description**

Comment of a Marker

**Usage**

```
comment(x)

comment(x) <- value
```

**Arguments**

x	A <a href="#">Marker</a> .
value	New comment.

**Value**

The comment string; the setter returns x.

**Examples**

```
m <- Marker("note", comment = "check color")
comment(m)
```

---

contains	<i>Does a TimeRange contain a time or range?</i>
----------	--

---

**Description**

For a RationalTime: start <= t < end\_exclusive. For a TimeRange: the other range lies strictly within on both ends (tolerance epsilon\_s).

**Usage**

```
contains(tr, other, epsilon_s = .EPS)
```

**Arguments**

tr	A TimeRange.
other	A RationalTime or TimeRange.
epsilon_s	Tolerance in seconds.

**Value**

Logical scalar.

**Examples**

```
tr <- TimeRange(RationalTime(0, 24), RationalTime(48, 24))
contains(tr, RationalTime(12, 24))
```

---

default_media_key	<i>The default media reference key ("DEFAULT_MEDIA")</i>
-------------------	--

---

**Description**

The default media reference key ("DEFAULT\_MEDIA")

**Usage**

```
default_media_key()
```

**Value**

The string "DEFAULT\_MEDIA".

**Examples**

```
default_media_key()
```

---

Effect	<i>Construct an OTIO effect</i>
--------	---------------------------------

---

## Description

Effect is a generic effect\_name plus metadata parameters. LinearTimeWarp is OTIO's linear speed change (time\_scalar).

## Usage

```
Effect(name = "", effect_name = "", enabled = TRUE, metadata = NULL)
```

```
LinearTimeWarp(name = "", effect_name = "", time_scalar = 1, enabled = TRUE,
               metadata = NULL)
```

```
TimeEffect(name = "", effect_name = "", metadata = NULL)
```

```
FreezeFrame(name = "", metadata = NULL)
```

## Arguments

name	Effect instance name (default empty).
effect_name	Effect kind label (default empty, matching OTIO).
enabled	Whether the effect is active (default TRUE).
metadata	Named list of parameters.
time_scalar	Linear time scale (playback rate multiplier).

## Value

An Effect.

## Examples

```
Effect("blur", "GaussianBlur", metadata = list(size = 4))
```

```
LinearTimeWarp("speed", effect_name = "LinearTimeWarp", time_scalar = 2)
```

---

effect_name	<i>Effect kind label</i>
-------------	--------------------------

---

**Description**

Effect kind label

**Usage**

```
effect_name(x)
```

```
effect_name(x) <- value
```

**Arguments**

x	An <a href="#">Effect</a> .
value	New effect_name.

**Value**

The effect kind string; the setter returns x.

**Examples**

```
e <- Effect("blur", "GaussianBlur")
effect_name(e)
```

---

effects	<i>Effects of an item or composition</i>
---------	--

---

**Description**

Effects of an item or composition

**Usage**

```
effects(x)
```

**Arguments**

x	An item or composition.
---	-------------------------

**Value**

A list of [Effect](#) objects.

**Examples**

```
clip <- add_effect(Clip("a"), FreezeFrame())
effects(clip)
```

---

enabled	<i>Whether an item, composition, or effect is enabled</i>
---------	---

---

**Description**

A disabled clip/track is muted; a disabled effect is bypassed.

**Usage**

```
enabled(x)
```

```
enabled(x) <- value
```

**Arguments**

x	An object with an enabled field.
value	TRUE or FALSE.

**Value**

TRUE if the object is enabled, else FALSE.

**Examples**

```
c1 <- Clip("a")
enabled(c1) <- FALSE
enabled(c1)
```

---

end_frame	<i>Last frame number of an ImageSequenceReference</i>
-----------	---

---

**Description**

Last frame number of an ImageSequenceReference

**Usage**

```
end_frame(x)
```

**Arguments**

x	An <a href="#">ImageSequenceReference</a> .
---	---

**Value**

Integer frame number of the last image in the sequence.

**Examples**

```
isr <- ImageSequenceReference("file:///frames", "frame.", ".exr", rate = 24,  
  available_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24)))  
end_frame(isr)
```

---

end_time_exclusive	<i>Exclusive / inclusive end of a TimeRange</i>
--------------------	---

---

**Description**

end\_time\_exclusive = start + duration. end\_time\_inclusive is the last whole frame; for a span of one frame or less it is the start time (matching opentime).

**Usage**

```
end_time_exclusive(x)
```

```
end_time_inclusive(x)
```

**Arguments**

x                   A TimeRange.

**Value**

A RationalTime.

**Examples**

```
tr <- TimeRange(RationalTime(0, 24), RationalTime(48, 24))  
end_time_exclusive(tr)  
end_time_inclusive(tr)
```

---

extended_by	<i>Smallest TimeRange covering two ranges</i>
-------------	---

---

**Description**

Smallest TimeRange covering two ranges

**Usage**

```
extended_by(tr, other)
```

**Arguments**

tr, other      TimeRanges.

**Value**

A TimeRange.

**Examples**

```
a <- TimeRange(RationalTime(0, 24), RationalTime(48, 24))
b <- TimeRange(RationalTime(24, 24), RationalTime(48, 24))
extended_by(a, b)
```

---

ExternalReference	<i>Construct a media reference</i>
-------------------	------------------------------------

---

**Description**

ExternalReference points a clip at media by URL. MissingReference is a placeholder for a clip with no resolvable media.

**Usage**

```
ExternalReference(target_url, available_range = NULL, metadata = NULL)
```

```
MissingReference(name = "", available_range = NULL, metadata = NULL)
```

**Arguments**

target_url	Media URL for an ExternalReference.
available_range	Optional <a href="#">TimeRange</a> of available media.
metadata	Named list of metadata.
name	Reference name (default empty).

**Value**

A media reference object.

**Examples**

```
ExternalReference("media/clip01.mp4")
MissingReference()
```

---

 fill

---

*Fill a gap with an item (3/4-point edit)*


---

**Description**

Replaces the gap covering `track_time` with `item`. The `reference_point` controls the transform: "Source" uses the clip's own duration, "Sequence" clamps to the gap, "Fit" time-warps the clip to fill the gap exactly. Mutates track in place.

**Usage**

```
fill(item, track, track_time, reference_point = "Source")
```

**Arguments**

<code>item</code>	The item to place (usually a clip).
<code>track</code>	A <a href="#">Track</a> .
<code>track_time</code>	A <a href="#">RationalTime</a> inside the gap to fill.
<code>reference_point</code>	One of "Source", "Sequence", "Fit".

**Value**

`track`, invisibly.

**Examples**

```
trk <- Track("V1")
append_child(trk, Gap(RationalTime(48, 24)))
cl <- Clip("a", source_range = TimeRange(RationalTime(0, 24),
                                         RationalTime(24, 24)))
fill(cl, trk, RationalTime(0, 24))
name(children(trk)[[1]])
```

---

find_clips	<i>All clips within an object (recursive)</i>
------------	---

---

**Description**

All clips within an object (recursive)

**Usage**

```
find_clips(x)
```

**Arguments**

x                    A Timeline, composition, or collection.

**Value**

A list of Clip objects.

**Examples**

```
trk <- Track("V1")
append_child(trk, Clip("a"))
length(find_clips(trk))
```

---

flatten_stack	<i>Flatten a stack of tracks into a single track</i>
---------------	--

---

**Description**

Composites top-down: starts from the topmost track and fills its holes (gaps and disabled items) with content from the tracks below, recursing downward. Transitions are preserved; a lower transition cut by a hole boundary errors with "Cannot trim in the middle of a transition" (matching OTIO).

**Usage**

```
flatten_stack(x)
```

**Arguments**

x                    A [Stack](#) or a list of [Tracks](#) (bottom-to-top).

**Value**

A flattened [Track](#).

**Examples**

```
trk <- add_child(Track("V1"), Clip("a",
  source_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24))))
flatten_stack(list(trk))
```

---

frame_for_time	<i>Frame for a time in an ImageSequenceReference</i>
----------------	--

---

**Description**

Errors if time falls outside the available range (matching OTIO).

**Usage**

```
frame_for_time(x, time)
```

**Arguments**

x	An <a href="#">ImageSequenceReference</a> .
time	A <a href="#">RationalTime</a> .

**Value**

Integer frame number for time.

**Examples**

```
isr <- ImageSequenceReference("file:///frames", "frame.", ".exr", rate = 24,
  available_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24)))
frame_for_time(isr, RationalTime(10, 24))
```

---

from_frames	<i>Construct a RationalTime from a frame number at a rate</i>
-------------	---

---

**Description**

Construct a [RationalTime](#) from a frame number at a rate

**Usage**

```
from_frames(frame, rate)
```

**Arguments**

frame	Integer frame number.
rate	Rate (fps).

**Value**

A RationalTime.

**Examples**

```
from_frames(48, 24)
```

---

from_json_file	<i>Read an OTIO JSON file into the object model</i>
----------------	---

---

**Description**

Read an OTIO JSON file into the object model

**Usage**

```
from_json_file(file_name)
```

**Arguments**

file\_name      Path to a .otio JSON file.

**Value**

The reconstructed OTIO object, with parent pointers wired.

**Examples**

```
f <- tempfile(fileext = ".otio")
to_json_file(Timeline("demo"), f)
name(from_json_file(f))
unlink(f)
```

---

from_json_string	<i>Parse an OTIO JSON string into the object model</i>
------------------	--

---

**Description**

Parse an OTIO JSON string into the object model

**Usage**

```
from_json_string(input)
```

**Arguments**

input            An OTIO JSON string.

**Value**

The reconstructed OTIO object (typically a [Timeline](#)), with parent pointers wired.

**Examples**

```
tl <- Timeline("demo")
identical(name(from_json_string(to_json_string(tl))), "demo")
```

---

from_seconds	<i>Construct a RationalTime from seconds at a rate</i>
--------------	--

---

**Description**

Construct a RationalTime from seconds at a rate

**Usage**

```
from_seconds(seconds, rate = 1)
```

**Arguments**

seconds	Numeric seconds.
rate	Rate (fps).

**Value**

A RationalTime.

**Examples**

```
from_seconds(1.5, 24)
```

---

from_time_string	<i>RationalTime from a time string ("HH:MM:SS.sss") at a rate</i>
------------------	---

---

**Description**

Preserves the fractional value (no rounding to a frame boundary).

**Usage**

```
from_time_string(time_string, rate)
```

**Arguments**

time_string	A time string.
rate	Rate (fps).

**Value**

A `RationalTime`.

**Examples**

```
from_time_string("00:00:03.75", 24)
```

---

from_timecode	<i>RationalTime from a SMPTE timecode</i>
---------------	---

---

**Description**

A ; frame separator is treated as drop-frame.

**Usage**

```
from_timecode(timecode, rate)
```

**Arguments**

timecode	A "HH:MM:SS:FF" (or ";"FF" for drop-frame) string.
rate	Timecode rate.

**Value**

A `RationalTime`.

**Examples**

```
from_timecode("01:00:00:00", 24)
```

---

Gap	<i>Construct a Gap</i>
-----	------------------------

---

**Description**

Empty space on a track of a given duration.

**Usage**

```
Gap(duration, name = "", metadata = NULL)
```

**Arguments**

duration	A <a href="#">RationalTime</a> .
name	Gap name.
metadata	Named list of metadata.

**Value**

A Gap.

**Examples**

```
Gap(RationalTime(15, 30))
```

---

<code>generator_kind</code>	<i>Generator kind of a GeneratorReference</i>
-----------------------------	---

---

**Description**

Generator kind of a GeneratorReference

**Usage**

```
generator_kind(x)  
generator_kind(x) <- value
```

**Arguments**

<code>x</code>	A <a href="#">GeneratorReference</a> .
<code>value</code>	New generator kind.

**Value**

The generator kind string; the setter returns `x`.

**Examples**

```
g <- GeneratorReference("bars", generator_kind = "SMPTEBars")  
generator_kind(g)
```

---

GeneratorReference      *Construct a GeneratorReference*

---

### Description

Procedurally-generated media (color bars, solids, ...).

### Usage

```
GeneratorReference(name = "", generator_kind = "", available_range = NULL,
                  parameters = NULL, metadata = NULL)
```

### Arguments

name	Reference name.
generator_kind	Generator kind (e.g. "SMPTEBars").
available_range	Optional <a href="#">TimeRange</a> .
parameters	Named list of generator parameters.
metadata	Named list of metadata.

### Value

A GeneratorReference.

### Examples

```
GeneratorReference("bars", generator_kind = "SMPTEBars")
```

---

global\_start\_time      *Global start time of a timeline*

---

### Description

Global start time of a timeline

### Usage

```
global_start_time(x)

global_start_time(x) <- value
```

### Arguments

x	A <a href="#">Timeline</a> .
value	A <a href="#">RationalTime</a> or NULL.

**Value**

The timeline's global start time (a [RationalTime](#) or NULL); the setter returns x.

**Examples**

```
tl <- Timeline("demo")
global_start_time(tl) <- RationalTime(0, 24)
global_start_time(tl)
```

---

has\_child

*Does a composition directly contain a child?*

---

**Description**

Does a composition directly contain a child?

**Usage**

```
has_child(x, child)
```

**Arguments**

x                    A composition or collection.  
child                An OTIO object.

**Value**

TRUE if child is a direct child of x, else FALSE.

**Examples**

```
trk <- Track("V1")
cl <- Clip("a")
append_child(trk, cl)
has_child(trk, cl)
```

---

has_clips	<i>Does an object contain any clips (recursive)?</i>
-----------	--

---

**Description**

Does an object contain any clips (recursive)?

**Usage**

```
has_clips(x)
```

**Arguments**

x                    A Timeline, composition, or collection.

**Value**

TRUE if x contains at least one clip, else FALSE.

**Examples**

```
trk <- Track("V1")
append_child(trk, Clip("a"))
has_clips(trk)
```

---

ImageSequenceReference

*Construct an ImageSequenceReference*

---

**Description**

A numbered image sequence (e.g. frame.0001.exr).

**Usage**

```
ImageSequenceReference(target_url_base = "", name_prefix = "",
                        name_suffix = "", start_frame = 1L, frame_step = 1L,
                        rate = 1, frame_zero_padding = 0L,
                        missing_frame_policy = "error", available_range = NULL,
                        metadata = NULL)
```

**Arguments**

target_url_base	Directory URL.
start_frame	First frame number (default 1).
frame_step	Frames between images (default 1).
rate	Frame rate (default 1).
frame_zero_padding	Zero-padding width for the frame number (default 0).
missing_frame_policy	One of "error", "hold", "black".
available_range	Optional <a href="#">TimeRange</a> .
metadata	Named list of metadata.
name_prefix, name_suffix	Filename prefix/suffix around the frame number.

**Value**

An ImageSequenceReference.

**Examples**

```
ImageSequenceReference("file:///frames", "frame.", ".exr",
                       frame_zero_padding = 4, rate = 24)
```

---

index_of_child	<i>1-based position of a child, or NA</i>
----------------	---

---

**Description**

1-based position of a child, or NA

**Usage**

```
index_of_child(x, child)
```

**Arguments**

x	A composition or collection.
child	An OTIO object.

**Value**

The 1-based integer position of child in x, or NA\_integer\_ if it is not a direct child.

**Examples**

```

trk <- Track("V1")
cl <- Clip("a")
append_child(trk, cl)
index_of_child(trk, cl)

```

---

insert

---

*Insert an item at a time, splitting the item it lands in*


---

**Description**

Inserts item at time, splitting whatever item spans that point; before the start it prepends, past the end it appends (filling any gap). Mutates composition in place.

**Usage**

```
insert(item, composition, time, remove_transitions = TRUE, fill_template = NULL)
```

**Arguments**

item	The item to insert (usually a clip).
composition	A <a href="#">Track</a> (or composition).
time	A <a href="#">RationalTime</a> to insert at.
remove_transitions	Remove transitions at time (default TRUE).
fill_template	Optional gap template when appending past the end.

**Value**

composition, invisibly.

**Examples**

```

trk <- Track("V1")
append_child(trk, Clip("a", source_range = TimeRange(RationalTime(0, 24),
                                                    RationalTime(48, 24))))
cl <- Clip("b", source_range = TimeRange(RationalTime(0, 24),
                                        RationalTime(12, 24)))
insert(cl, trk, RationalTime(24, 24))
length(children(trk))

```

---

insert_child	<i>Insert a child at a 1-based position (in place)</i>
--------------	--

---

**Description**

Insert a child at a 1-based position (in place)

**Usage**

```
insert_child(x, index, child)
```

**Arguments**

x	A composition or collection.
index	1-based position.
child	An OTIO object with no parent.

**Value**

x, invisibly.

**Examples**

```
trk <- Track("V1")
append_child(trk, Clip("b"))
insert_child(trk, 1, Clip("a"))
name(children(trk)[[1]])
```

---

intersects	<i>Do two TimeRanges intersect?</i>
------------	-------------------------------------

---

**Description**

Do two TimeRanges intersect?

**Usage**

```
intersects(tr, other, epsilon_s = .EPS)
```

**Arguments**

epsilon_s	Tolerance in seconds.
tr, other	TimeRanges.

**Value**

Logical scalar.

**Examples**

```
a <- TimeRange(RationalTime(0, 24), RationalTime(48, 24))
b <- TimeRange(RationalTime(24, 24), RationalTime(48, 24))
intersects(a, b)
```

---

is\_effect

*Is x an Effect?*

---

**Description**

Is x an Effect?

**Usage**

```
is_effect(x)
```

**Arguments**

x                    Object to test.

**Value**

TRUE if x is an Effect, else FALSE.

**Examples**

```
is_effect(Effect("blur", "GaussianBlur"))
```

---

is\_equivalent\_to

*Are two OTIO objects equivalent?*

---

**Description**

Structural equality, compared via canonical OTIO JSON.

**Usage**

```
is_equivalent_to(x, other)
```

**Arguments**

x, other            OTIO objects.

**Value**

Logical scalar.

**Examples**

```
is_equivalent_to(Timeline("a"), Timeline("a"))
```

---

*is\_missing\_reference*    *Is x a MissingReference?*

---

**Description**

Is x a MissingReference?

**Usage**

```
is_missing_reference(x)
```

**Arguments**

x                    Object to test.

**Value**

TRUE if x is a MissingReference, else FALSE.

**Examples**

```
is_missing_reference(MissingReference())
```

---

*is\_otio*                    *Type predicates for OTIO objects*

---

**Description**

Type predicates for OTIO objects

**Usage**

```
is_otio(x)
```

```
is_timeline(x)
```

```
is_composition(x)
```

```
is_media_reference(x)
```

**Arguments**

x                    Object to test.

**Value**

TRUE if x is of the corresponding class, else FALSE.

**Examples**

```
is_otio(Timeline("demo"))
is_timeline(Track("V1"))
```

---

is_parent_of	<i>Is x an ancestor of other?</i>
--------------	-----------------------------------

---

**Description**

Is x an ancestor of other?

**Usage**

```
is_parent_of(x, other)
```

**Arguments**

x                    A composition or collection.  
 other                An OTIO object.

**Value**

TRUE if x appears anywhere in other's parent chain, else FALSE.

**Examples**

```
trk <- Track("V1")
cl <- Clip("a")
append_child(trk, cl)
is_parent_of(trk, cl)
```

---

is_rational_time	<i>Is x a RationalTime / TimeRange?</i>
------------------	---

---

**Description**

Is x a RationalTime / TimeRange?

**Usage**

```
is_rational_time(x)
```

```
is_time_range(x)
```

**Arguments**

x                    Object to test.

**Value**

Logical scalar.

**Examples**

```
is_rational_time(RationalTime(24, 24))  
is_time_range(TimeRange(RationalTime(0, 24), RationalTime(48, 24)))
```

---

is_unknown_schema	<i>Is an object's schema unknown?</i>
-------------------	---------------------------------------

---

**Description**

TRUE when the object's schema type is not a recognised OTIO type (e.g. parsed from JSON written by a newer or third-party schema).

**Usage**

```
is_unknown_schema(x)
```

**Arguments**

x                    An OTIO object.

**Value**

A logical scalar.

**Examples**

```
is_unknown_schema(Timeline("demo"))
```

---

Item	<i>Construct a base Item</i>
------	------------------------------

---

### Description

A generic OTIO Item (the base class of clips, gaps, and compositions). Mainly used by the "Fit" reference point of [fill](#), which wraps media in a plain item carrying a time warp.

### Usage

```
Item(name = "", source_range = NULL, effects = NULL, markers = NULL,
      enabled = TRUE, metadata = NULL)
```

### Arguments

name	Item name.
source_range	Optional <a href="#">TimeRange</a> .
effects	List of <a href="#">Effects</a> .
markers	List of <a href="#">Markers</a> .
enabled	Whether the item is enabled (default TRUE).
metadata	Named list of metadata.

### Value

An Item.

### Examples

```
Item("warp", source_range = TimeRange(RationalTime(0, 30), RationalTime(60, 30)))
```

---

kind	<i>Get or set a track's kind</i>
------	----------------------------------

---

### Description

Get or set a track's kind

### Usage

```
kind(x)
```

```
kind(x) <- value
```

**Arguments**

x                    A [Track](#).  
value                "Video" or "Audio".

**Value**

The track kind, "Video" or "Audio".

**Examples**

```
trk <- Track("A1", kind = "Audio")  
kind(trk)
```

---

marked_range	<i>Marked range of a Marker</i>
--------------	---------------------------------

---

**Description**

Marked range of a Marker

**Usage**

```
marked_range(x)  
marked_range(x) <- value
```

**Arguments**

x                    A [Marker](#).  
value                A [TimeRange](#).

**Value**

The marker's [TimeRange](#); the setter returns x.

**Examples**

```
m <- Marker("note", TimeRange(RationalTime(0, 30), RationalTime(30, 30)))  
marked_range(m)
```

---

Marker	<i>Construct a Marker</i>
--------	---------------------------

---

**Description**

Annotates a time range on an item.

**Usage**

```
Marker(name = "", marked_range = NULL, color = "GREEN", comment = "",
        metadata = NULL)
```

**Arguments**

name	Marker name.
marked_range	A <a href="#">TimeRange</a> .
color	Marker color (default "GREEN").
comment	Free-text comment.
metadata	Named list of metadata.

**Value**

A Marker.

**Examples**

```
Marker("note", TimeRange(RationalTime(0, 30), RationalTime(30, 30)),
        comment = "check color")
```

---

media_reference	<i>Active media reference of a clip</i>
-----------------	---

---

**Description**

Active media reference of a clip

**Usage**

```
media_reference(x)

media_reference(x) <- value
```

**Arguments**

x	A <a href="#">Clip</a> .
value	A media reference.

**Value**

The clip's active media reference object.

**Examples**

```
cl <- Clip("a", ExternalReference("a.mp4"))
media_reference(cl)
```

---

media_references	<i>Media references of a clip</i>
------------------	-----------------------------------

---

**Description**

Media references of a clip

**Usage**

```
media_references(x)
```

**Arguments**

x                    A [Clip](#).

**Value**

The clip's named list of media references.

**Examples**

```
cl <- Clip("a", ExternalReference("a.mp4"))
names(media_references(cl))
```

---

MediaReference	<i>Construct a generic MediaReference</i>
----------------	---

---

**Description**

Construct a generic MediaReference

**Usage**

```
MediaReference(name = "", available_range = NULL, metadata = NULL)
```

**Arguments**

name	Reference name.
available_range	Optional <a href="#">TimeRange</a> .
metadata	Named list of metadata.

**Value**

A `MediaReference`.

**Examples**

```
MediaReference("ref")
```

---

metadata	<i>Get or set object metadata</i>
----------	-----------------------------------

---

**Description**

Get or set object metadata

**Usage**

```
metadata(x)
metadata(x) <- value
```

**Arguments**

x	An OTIO object.
value	A named list.

**Value**

The object's metadata as a named list (possibly empty).

**Examples**

```
c1 <- Clip("a")
metadata(c1) <- list(note = "take 2")
metadata(c1)
```

---

name	<i>Get or set the name of an OTIO object</i>
------	--

---

**Description**

Get or set the name of an OTIO object

**Usage**

```
name(x)
```

```
name(x) <- value
```

**Arguments**

x	An OTIO object.
value	New name.

**Value**

The object's name, a character string.

**Examples**

```
cl <- Clip("a")  
name(cl) <- "b"  
name(cl)
```

---

number_of_images_in_sequence	<i>Number of images in an ImageSequenceReference</i>
------------------------------	--

---

**Description**

Number of images in an ImageSequenceReference

**Usage**

```
number_of_images_in_sequence(x)
```

**Arguments**

x	An <a href="#">ImageSequenceReference</a> .
---	---

**Value**

Integer count of images (0 if no available range is set).

**Examples**

```
isr <- ImageSequenceReference("file:///frames", "frame.", ".exr", rate = 24,
  available_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24)))
number_of_images_in_sequence(isr)
```

---

overlapping

*Does an item overlap its neighbours?*


---

**Description**

Only transitions overlap (they span a cut); everything else returns FALSE.

**Usage**

```
overlapping(x)
```

**Arguments**

x                    An item.

**Value**

Logical scalar.

**Examples**

```
overlapping(Clip("a", ExternalReference("a.mp4")))
```

---

overlaps

*Do two TimeRanges overlap (cross without containment)?*


---

**Description**

Do two TimeRanges overlap (cross without containment)?

**Usage**

```
overlaps(tr, other, epsilon_s = .EPS)
```

**Arguments**

epsilon\_s            Tolerance in seconds.  
tr, other            TimeRanges.

**Value**

Logical scalar.

**Examples**

```
a <- TimeRange(RationalTime(0, 24), RationalTime(48, 24))
b <- TimeRange(RationalTime(24, 24), RationalTime(48, 24))
overlaps(a, b)
```

---

 overwrite

*Overwrite a span of a composition with an item*


---

**Description**

Places `item` over `range` (in composition coordinates), partitioning or removing the items it covers and filling any hole before/after with a gap (or `fill_template`). Mutates composition in place.

**Usage**

```
overwrite(item, composition, range, remove_transitions = TRUE,
          fill_template = NULL)
```

**Arguments**

`item`            The item to place (usually a clip).  
`composition`    A [Track](#) (or composition).  
`range`            A [TimeRange](#) to overwrite.  
`remove_transitions`  
                   Remove transitions within range (default TRUE).  
`fill_template`   Optional gap template for holes.

**Value**

`composition`, invisibly.

**Examples**

```
trk <- Track("V1")
append_child(trk, Gap(RationalTime(48, 24)))
cl <- Clip("a", source_range = TimeRange(RationalTime(0, 24),
                                         RationalTime(12, 24)))
overwrite(cl, trk, TimeRange(RationalTime(12, 24), RationalTime(12, 24)))
length(children(trk))
```

parameters

*Parameters of a GeneratorReference*

---

**Description**

Parameters of a GeneratorReference

**Usage**

```
parameters(x)
```

```
parameters(x) <- value
```

**Arguments**

x	A <a href="#">GeneratorReference</a> .
value	A named list of parameters.

**Value**

A named list of generator parameters; the setter returns x.

**Examples**

```
g <- GeneratorReference("bars", parameters = list(width = 1920))
parameters(g)
```

---

parent

*Parent of an OTIO object*

---

**Description**

The containing composition/collection, or NULL. A Timeline's root track Stack is parentless.

**Usage**

```
parent(x)
```

**Arguments**

x	An OTIO object.
---	-----------------

**Value**

The parent object or NULL.

**Examples**

```

trk <- Track("V1")
cl <- Clip("a")
append_child(trk, cl)
name(parent(cl))

```

---

presentation\_time\_for\_image\_number

*Presentation time of the nth image (1-based)*

---

**Description**

Presentation time of the nth image (1-based)

**Usage**

```
presentation_time_for_image_number(x, image_number)
```

**Arguments**

x                    An [ImageSequenceReference](#).  
image\_number        1-based image index.

**Value**

A [RationalTime](#) at the reference's rate.

**Examples**

```

isr <- ImageSequenceReference("file:///frames", "frame.", ".exr", rate = 24,
  available_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24)))
presentation_time_for_image_number(isr, 2)

```

---

range\_from\_start\_end\_time

*Construct a TimeRange from start and exclusive end times*

---

**Description**

Duration is computed in the start time's rate (opentime convention).

**Usage**

```
range_from_start_end_time(start_time, end_time_exclusive)
```

**Arguments**

start\_time      A RationalTime.  
 end\_time\_exclusive  
                   A RationalTime.

**Value**

A TimeRange.

**Examples**

```
range_from_start_end_time(RationalTime(0, 24), RationalTime(48, 24))
```

---

range_in_parent	<i>Range of an item within its parent composition</i>
-----------------	---

---

**Description**

Computed from the durations of the preceding siblings (gaps included). The duration is the item's trimmed duration.

**Usage**

```
range_in_parent(x)
```

**Arguments**

x                    An item with a parent.

**Value**

A [TimeRange](#) in the parent's coordinates.

**Examples**

```
trk <- Track("V1")
cl <- Clip("a",
  source_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24)))
append_child(trk, cl)
range_in_parent(cl)
```

---

RationalTime	<i>Construct a RationalTime</i>
--------------	---------------------------------

---

**Description**

A value / rate pair measured in the rate's units (frame number / fps for frame-based work).

**Usage**

```
RationalTime(value, rate = 1)
```

**Arguments**

value	Time value. Coerced to double.
rate	Rate (fps); must be > 0. Default 1.

**Value**

A RationalTime.

**Examples**

```
RationalTime(180, 30)
```

---

read_otiod	<i>Read an OTIOD bundle</i>
------------	-----------------------------

---

**Description**

Reads <dir>/content.otio into the object model. Media references stay relative to the bundle directory.

**Usage**

```
read_otiod(dir)
```

**Arguments**

dir	Bundle directory.
-----	-------------------

**Value**

The bundle's OTIO object (typically a [Timeline](#)).

**Examples**

```
d <- tempfile("bundle")
write_otiod(Timeline("demo"), d)
name(read_otiod(d))
unlink(d, recursive = TRUE)
```

---

register\_downgrade\_function

*Register a schema downgrade function*

---

**Description**

fn receives the field list of a schema\_name object at version\_to\_downgrade\_from and returns the field list one version lower. Stored for use when writing older schema versions.

**Usage**

```
register_downgrade_function(schema_name, version_to_downgrade_from, fn)
```

**Arguments**

schema\_name      Schema type name. Must be a known OTIO schema.

version\_to\_downgrade\_from  
                  Source version (integer).

fn                Function of one argument (the field list) returning a field list.

**Value**

TRUE if registered; FALSE for an unknown schema or a duplicate (schema, version) pair (matching RcppOTIO).

**Examples**

```
register_downgrade_function("Effect", 2, function(fields) fields)
```

---

 register\_upgrade\_function

*Register a schema upgrade function*


---

### Description

fn receives the parsed field list of a schema\_name object at the version below version\_to\_upgrade\_to and returns the upgraded list. It is applied automatically by [from\\_json\\_string](#) when an older version is read.

### Usage

```
register_upgrade_function(schema_name, version_to_upgrade_to, fn)
```

### Arguments

schema_name	Schema type name (e.g. "Marker"). Must be a known OTIO schema.
version_to_upgrade_to	Target version (integer).
fn	Function of one argument (the field list) returning a field list.

### Value

TRUE if registered; FALSE for an unknown schema or a duplicate (schema, version) pair (matching RcppOTIO).

### Examples

```
register_upgrade_function("Effect", 2, function(fields) fields)
```

---

 remove

*Remove the item at a time, optionally leaving a gap*


---

### Description

Removes the item covering time; if fill, replaces it with a gap (or fill\_template) of the same range, otherwise the neighbours concatenate. Mutates composition in place.

### Usage

```
remove(composition, time, fill = TRUE, fill_template = NULL)
```

**Arguments**

composition    A [Track](#) (or composition).  
time            A [RationalTime](#) within the item to remove.  
fill             Fill the hole with a gap/template (default TRUE).  
fill\_template   Optional item to fill with (default a gap).

**Value**

composition, invisibly.

**Examples**

```
trk <- Track("V1")
append_child(trk, Clip("a", source_range = TimeRange(RationalTime(0, 24),
                                                    RationalTime(48, 24))))
remove(trk, RationalTime(0, 24))
class(children(trk)[[1]])
```

---

remove_child	<i>Remove the child at a position (in place)</i>
--------------	--

---

**Description**

Remove the child at a position (in place)

**Usage**

```
remove_child(x, index)
```

**Arguments**

x                A composition or collection.  
index            1-based position.

**Value**

x, invisibly.

**Examples**

```
trk <- Track("V1")
append_child(trk, Clip("a"))
remove_child(trk, 1)
length(children(trk))
```

---

rescaled_to	<i>Rescale a RationalTime to a new rate</i>
-------------	---

---

**Description**

Rescale a RationalTime to a new rate

**Usage**

```
rescaled_to(x, new_rate)
```

**Arguments**

x	A RationalTime.
new_rate	Target rate (fps).

**Value**

A RationalTime at new\_rate.

**Examples**

```
rescaled_to(RationalTime(24, 24), 48)
```

---

ripple	<i>Ripple an item's source range</i>
--------	--------------------------------------

---

**Description**

Adjusts the item's source\_range start by delta\_in and exclusive end by delta\_out without moving any other item. Mutates in place.

**Usage**

```
ripple(item, delta_in, delta_out)
```

**Arguments**

item	An item (usually a clip).
delta_in, delta_out	<a href="#">RationalTime</a> adjustments to the start and exclusive end.

**Value**

item, invisibly.

**Examples**

```
cl <- Clip("a", source_range = TimeRange(RationalTime(0, 24),
                                         RationalTime(48, 24)))
ripple(cl, RationalTime(0, 24), RationalTime(12, 24))
source_range(cl)
```

---

roll

*Roll an item, adjusting adjacent items to fit*


---

**Description**

Adjusts the item's start by `delta_in` and exclusive end by `delta_out`, absorbing the change into the neighbouring items' source ranges (no new items are created); clamps to media available ranges. Mutates in place.

**Usage**

```
roll(item, delta_in, delta_out)
```

**Arguments**

`item`            An item (usually a clip).  
`delta_in, delta_out`    `RationalTime` adjustments.

**Value**

`item`, invisibly.

**Examples**

```
trk <- Track("V1")
a <- Clip("a", source_range = TimeRange(RationalTime(0, 24),
                                         RationalTime(24, 24)))
b <- Clip("b", source_range = TimeRange(RationalTime(24, 24),
                                         RationalTime(24, 24)))
append_child(trk, a)
append_child(trk, b)
roll(b, RationalTime(-6, 24), RationalTime(0, 24))
```

---

schema_name	<i>Schema name of an OTIO object</i>
-------------	--------------------------------------

---

**Description**

The schema type (without the version suffix); unrecognised types report "UnknownSchema" (matching OTIO).

**Usage**

```
schema_name(x)
```

**Arguments**

x                    An OTIO object.

**Value**

A character scalar.

**Examples**

```
schema_name(Clip("shot1"))
```

---

schema_version	<i>Schema version of an OTIO object</i>
----------------	---

---

**Description**

Schema version of an OTIO object

**Usage**

```
schema_version(x)
```

**Arguments**

x                    An OTIO object.

**Value**

An integer.

**Examples**

```
schema_version(Clip("shot1"))
```

---

 SerializableCollection

*Construct a SerializableCollection*


---

**Description**

A flat, named collection of OTIO objects (not a composition).

**Usage**

```
SerializableCollection(name = "", children = list(), metadata = NULL)
```

**Arguments**

name	Collection name.
children	A list of OTIO objects.
metadata	Named list of metadata.

**Value**

A SerializableCollection.

**Examples**

```
col <- SerializableCollection("shots", list(Clip("a"), Clip("b")))
length(children(col))
```

---

 set\_child

*Replace the child at a position (in place)*


---

**Description**

Detaches the replaced child and attaches the new one. Returns x invisibly.

**Usage**

```
set_child(x, index, child)
```

**Arguments**

x	A composition or collection.
index	1-based position.
child	An OTIO object with no parent.

**Value**

x, invisibly.

**Examples**

```
trk <- Track("V1")
append_child(trk, Clip("a"))
set_child(trk, 1, Clip("b"))
name(children(trk)[[1]])
```

---

set_children	<i>Replace all children (in place)</i>
--------------	--

---

**Description**

Available as `set_children(x, kids)` and `set_children(x) <- kids`.

**Usage**

```
set_children(x, children)
```

```
set_children(x) <- value
```

**Arguments**

x                   A composition or collection.

children, value    A list of parentless OTIO objects.

**Value**

x (invisibly from the function form).

**Examples**

```
trk <- Track("V1")
set_children(trk, list(Clip("a"), Clip("b")))
length(children(trk))
```

---

set\_media\_references *Replace the media references of a clip (in place)*

---

### Description

Replace the media references of a clip (in place)

### Usage

```
set_media_references(x, media_references, new_active_key = NULL)
```

### Arguments

x [A Clip](#).  
 media\_references [A named list of media references](#).  
 new\_active\_key [Optional new active key](#).

### Value

x, invisibly.

### Examples

```
cl <- Clip("a")
set_media_references(cl, list(DEFAULT_MEDIA = ExternalReference("a.mp4")))
target_url(cl)
```

---

slice *Slice an item in two at a time*

---

### Description

Splits the item covering time into two adjacent items at that point. Mutates composition in place. A slice exactly at an item boundary is a no-op; slicing through a transition removes it (or errors if `remove_transitions` is FALSE).

### Usage

```
slice(composition, time, remove_transitions = TRUE)
```

### Arguments

composition [A Track](#) (or composition).  
 time [A RationalTime](#) to slice at.  
 remove\_transitions [Remove transitions at time \(default TRUE\)](#).

**Value**

composition, invisibly.

**Examples**

```

trk <- Track("V1")
append_child(trk, Clip("a", source_range = TimeRange(RationalTime(0, 24),
                                                    RationalTime(48, 24))))
slice(trk, RationalTime(24, 24))
length(children(trk))

```

---

slide

*Slide an item, adjusting the previous item's duration*

---

**Description**

Moves the item's start by delta by changing the previous item's duration; the item's own source range is unchanged. No-op for the first item. Mutates in place.

**Usage**

```
slide(item, delta)
```

**Arguments**

`item`            An item (usually a clip).  
`delta`            A [RationalTime](#) to slide by.

**Value**

`item`, invisibly.

**Examples**

```

trk <- Track("V1")
append_child(trk, Gap(RationalTime(24, 24)))
cl <- Clip("a", source_range = TimeRange(RationalTime(0, 24),
                                        RationalTime(48, 24)))
append_child(trk, cl)
slide(cl, RationalTime(-6, 24))

```

---

slip *Slip an item's source range*

---

### Description

Shifts the item's `source_range` start by `delta` without changing its duration or any surrounding item; clamps to the media available range when present. Mutates `item` in place.

### Usage

```
slip(item, delta)
```

### Arguments

`item` An item (usually a clip).  
`delta` A [RationalTime](#) to slip by.

### Value

`item`, invisibly.

### Examples

```
cl <- Clip("a", source_range = TimeRange(RationalTime(0, 24),
                                         RationalTime(48, 24)))
slip(cl, RationalTime(6, 24))
source_range(cl)
```

---

source\_range *Get or set an item's source range*

---

### Description

Get or set an item's source range

### Usage

```
source_range(x)

source_range(x) <- value
```

### Arguments

`x` An item (clip, gap, track).  
`value` A [TimeRange](#).

**Value**

The item's [TimeRange](#), or NULL if unset.

**Examples**

```
cl <- Clip("a")
source_range(cl) <- TimeRange(RationalTime(0, 24), RationalTime(48, 24))
source_range(cl)
```

---

Stack

*Construct a Stack*

---

**Description**

Parallel tracks (a Timeline's tracks is a Stack).

**Usage**

```
Stack(name = "tracks", source_range = NULL, metadata = NULL)
```

**Arguments**

name	Stack name (default "tracks").
source_range	Optional <a href="#">TimeRange</a> .
metadata	Named list of metadata.

**Value**

A Stack.

**Examples**

```
Stack()
```

---

start_time	<i>TimeRange start_time and duration</i>
------------	--

---

**Description**

TimeRange start\_time and duration

**Usage**

```
start_time(x)
```

```
duration(x)
```

**Arguments**

x                   A TimeRange.

**Value**

A RationalTime.

**Examples**

```
tr <- TimeRange(RationalTime(0, 24), RationalTime(48, 24))
start_time(tr)
duration(tr)
```

---

target_url	<i>Target URL of a clip or external reference</i>
------------	---

---

**Description**

Target URL of a clip or external reference

**Usage**

```
target_url(x)
```

```
target_url(x) <- value
```

**Arguments**

x                   A [Clip](#) or [ExternalReference](#).

value               New URL.

**Value**

The target URL string, or NULL if the active reference has no URL.

**Examples**

```
cl <- Clip("a", ExternalReference("a.mp4"))
target_url(cl) <- "b.mp4"
target_url(cl)
```

---

target_url_base	<i>ImageSequenceReference fields</i>
-----------------	--------------------------------------

---

**Description**

ImageSequenceReference fields

**Usage**

```
target_url_base(x)

target_url_base(x) <- value

name_prefix(x)

name_suffix(x)

start_frame(x)

frame_step(x)

frame_zero_padding(x)
```

**Arguments**

x	An <a href="#">ImageSequenceReference</a> .
value	New value.

**Value**

The stored field value; the setter returns x.

**Examples**

```
isr <- ImageSequenceReference("file:///frames", "frame.", ".exr",
                             frame_zero_padding = 4, rate = 24)
target_url_base(isr)
start_frame(isr)
```

---

target\_url\_for\_image\_number  
*Target URL of the nth image (1-based)*

---

**Description**

Target URL of the nth image (1-based)

**Usage**

```
target_url_for_image_number(x, image_number)
```

**Arguments**

x                    An [ImageSequenceReference](#).  
 image\_number       1-based image index.

**Value**

The image URL as a string.

**Examples**

```
isr <- ImageSequenceReference("file:///frames", "frame.", ".exr",
  frame_zero_padding = 4, rate = 24,
  available_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24)))
target_url_for_image_number(isr, 1)
```

---

time\_scalar                *Time scalar of a LinearTimeWarp*

---

**Description**

Time scalar of a LinearTimeWarp

**Usage**

```
time_scalar(x)

time_scalar(x) <- value
```

**Arguments**

x                    A [LinearTimeWarp](#).  
 value               New time scalar.

**Value**

The numeric time scalar; the setter returns x.

**Examples**

```
w <- LinearTimeWarp(effect_name = "LinearTimeWarp", time_scalar = 2)
time_scalar(w)
```

---

Timeline

*Construct a Timeline*

---

**Description**

Wraps a [Stack](#) of tracks. Add tracks with [add\\_track](#).

**Usage**

```
Timeline(name = "", global_start_time = NULL, metadata = NULL)
```

**Arguments**

name	Timeline name.
global_start_time	Optional <a href="#">RationalTime</a> .
metadata	Named list of metadata.

**Value**

A Timeline.

**Examples**

```
Timeline("demo")
```

---

TimeRange                      *Construct a TimeRange*

---

**Description**

A start\_time plus a duration, both [RationalTime](#).

**Usage**

```
TimeRange(start_time, duration)
```

**Arguments**

start_time	A RationalTime.
duration	A RationalTime.

**Value**

A TimeRange.

**Examples**

```
TimeRange(RationalTime(0, 30), RationalTime(180, 30))
```

---

TimeTransform                      *Construct a TimeTransform*

---

**Description**

An offset/scale/rate transform (OTIO TimeTransform). Note: in RcppOTIO TimeTransform is a plain opentime value type that does not serialize to JSON; rotio gives it an OTIO\_SCHEMA for its own (de)serialization, which is not verified against RcppOTIO JSON.

**Usage**

```
TimeTransform(offset = RationalTime(0, 1), scale = 1, rate = -1)
```

**Arguments**

offset	A RationalTime offset (default 0).
scale	Time scale (default 1).
rate	Target rate, or -1 to preserve (default -1).

**Value**

A TimeTransform.

**Examples**

```
TimeTransform(RationalTime(12, 24), scale = 2)
```

---

to_frames	<i>Frame number of a RationalTime</i>
-----------	---------------------------------------

---

**Description**

Integer frame number. With no rate, truncates at the time's own rate; with a rate, rescales first. Truncates toward zero (opentime int(value\_rescaled\_to(rate))).

**Usage**

```
to_frames(x, rate = NULL)
```

**Arguments**

x	A RationalTime.
rate	Optional target rate to rescale to first.

**Value**

Integer scalar, the frame number.

**Examples**

```
to_frames(from_seconds(1.5, 24))
to_frames(RationalTime(48, 24), 48)
```

---

to_json_file	<i>Write an OTIO object to a JSON file</i>
--------------	--

---

**Description**

Write an OTIO object to a JSON file

**Usage**

```
to_json_file(x, file_name, indent = 2, target_schema_versions = NULL)
```

**Arguments**

x	An OTIO object (typically a <a href="#">Timeline</a> ).
file_name	Output path (conventionally content.otio).
indent	Indent width (default 2).
target_schema_versions	Optional schema downgrade targets; see <a href="#">to_json_string</a> .

**Value**

file\_name, invisibly.

**Examples**

```
f <- tempfile(fileext = ".otio")
to_json_file(Timeline("demo"), f)
unlink(f)
```

---

to_json_string	<i>Serialize an OTIO object to a JSON string</i>
----------------	--

---

**Description**

Emits canonical OpenTimelineIO JSON. Each object carries its own OTIO\_SCHEMA, so the output is the standard .otio format and parses in any OTIO reader (verify with [validate\\_with\\_RcppOTIO](#)).

**Usage**

```
to_json_string(x, indent = 2, target_schema_versions = NULL)
```

**Arguments**

x	An OTIO object (typically a <a href="#">Timeline</a> ).
indent	Indent width for pretty-printing (default 2). Use 0 for compact.
target_schema_versions	Optional named integer vector mapping schema type to a target version; objects above that version are downgraded using the registered downgrade functions (e.g. c(Clip = 1L)).

**Value**

A JSON string.

**Examples**

```
to_json_string(Timeline("demo"))
```

---

to_seconds	<i>Convert a RationalTime to seconds</i>
------------	--

---

**Description**

Convert a RationalTime to seconds

**Usage**

```
to_seconds(x)
```

**Arguments**

x                   A RationalTime.

**Value**

Numeric scalar, the time in seconds.

**Examples**

```
to_seconds(RationalTime(48, 24))
```

---

to_time_string	<i>Time string ("HH:MM:SS.sss") for a RationalTime</i>
----------------	--

---

**Description**

Time string ("HH:MM:SS.sss") for a RationalTime

**Usage**

```
to_time_string(x)
```

**Arguments**

x                   A RationalTime.

**Value**

Character scalar, the time as "HH:MM:SS.sss".

**Examples**

```
to_time_string(RationalTime(90, 24))
```

---

to_timecode	<i>SMPTE timecode for a RationalTime</i>
-------------	--

---

**Description**

SMPTE timecode for a RationalTime

**Usage**

```
to_timecode(x, rate = NULL, drop_frame = NULL)
```

**Arguments**

x	A RationalTime.
rate	Timecode rate (default x's rate).
drop_frame	Drop-frame timecode. NULL (default) infers it from the rate (on for 30000/1001 and 60000/1001), matching opentime's InferFromRate; pass TRUE/FALSE to force.

**Value**

Character scalar, the timecode as "HH:MM:SS:FF" (or ";FF" for drop-frame).

**Examples**

```
to_timecode(RationalTime(86400, 24))
```

---

Track	<i>Construct a Track</i>
-------	--------------------------

---

**Description**

An ordered sequence of items. Add children with [add\\_child](#) (or the mutating [append\\_child](#)).

**Usage**

```
Track(name, kind = "Video", source_range = NULL, metadata = NULL)
```

**Arguments**

name	Track name.
kind	"Video" (default) or "Audio".
source_range	Optional <a href="#">TimeRange</a> .
metadata	Named list of metadata.

**Value**

A [Track](#).

**Examples**

```
Track("V1", kind = "Video")
```

---

track\_trimmed\_to\_range

*Trim a track to a time range*

---

**Description**

Returns a new track containing each child trimmed to the portion that falls within `trim_range` (in track coordinates); children fully outside are dropped.

**Usage**

```
track_trimmed_to_range(in_track, trim_range)
```

**Arguments**

`in_track`      A [Track](#).

`trim_range`    A [TimeRange](#) in track coordinates.

**Value**

A new [Track](#).

**Examples**

```
trk <- add_child(Track("V1"), Clip("a",
  source_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24))))
track_trimmed_to_range(trk,
  TimeRange(RationalTime(0, 24), RationalTime(24, 24)))
```

---

tracks	<i>The track stack of a timeline</i>
--------	--------------------------------------

---

**Description**

The track stack of a timeline

**Usage**

```
tracks(x)
```

```
tracks(x) <- value
```

**Arguments**

x	A <a href="#">Timeline</a> .
value	A <a href="#">Stack</a> .

**Value**

The timeline's [Stack](#).

**Examples**

```
t1 <- add_track(Timeline("demo"), Track("V1", kind = "Video"))
length(children(tracks(t1)))
```

---

Transition	<i>Construct a Transition</i>
------------	-------------------------------

---

**Description**

A transition (e.g. a dissolve) between adjacent items on a track.

**Usage**

```
Transition(name = "", transition_type = "", in_offset = RationalTime(0, 1),
           out_offset = RationalTime(0, 1), metadata = NULL)
```

**Arguments**

name	Transition name.
transition_type	Transition type (e.g. "SMPTE_Dissolve").
metadata	Named list of metadata.
in_offset, out_offset	<a href="#">RationalTime</a> offsets into the neighbouring clips.

**Value**

A Transition.

**Examples**

```
Transition(transition_type = "SMPTE_Dissolve",
          in_offset = RationalTime(5, 30), out_offset = RationalTime(5, 30))
```

---

transition_type	<i>Transition type / offsets</i>
-----------------	----------------------------------

---

**Description**

Transition type / offsets

**Usage**

```
transition_type(x)
transition_type(x) <- value
in_offset(x)
in_offset(x) <- value
out_offset(x)
out_offset(x) <- value
```

**Arguments**

x	A <a href="#">Transition</a> .
value	New value.

**Value**

transition\_type returns a string, in\_offset and out\_offset a [RationalTime](#); setters return x.

**Examples**

```
tr <- Transition(transition_type = "SMPTE_Dissolve",
                in_offset = RationalTime(5, 30), out_offset = RationalTime(5, 30))
transition_type(tr)
in_offset(tr)
```

---

trim	<i>Trim an item, filling the freed time with gap</i>
------	--

---

### Description

Adjusts the item's start by `delta_in` (also extending the previous item) and its exclusive end by `delta_out`, filling now-empty time with a gap (or `fill_template`); an adjacent gap is grown instead. Mutates in place.

### Usage

```
trim(item, delta_in, delta_out, fill_template = NULL)
```

### Arguments

`item` An item (usually a clip).  
`fill_template` Optional item to fill freed time with (default a gap).  
`delta_in, delta_out` [RationalTime](#) adjustments.

### Value

`item`, invisibly.

### Examples

```
trk <- Track("V1")
cl <- Clip("a", source_range = TimeRange(RationalTime(0, 24),
                                         RationalTime(48, 24)))
append_child(trk, cl)
trim(cl, RationalTime(6, 24), RationalTime(0, 24))
source_range(cl)
```

---

trimmed_range	<i>Trimmed range of an item</i>
---------------	---------------------------------

---

### Description

The item's `source_range` if set, else (for a clip) the active media reference's available range.

### Usage

```
trimmed_range(x)
```

### Arguments

`x` An item (clip, gap).

**Value**

A [TimeRange](#).

**Examples**

```
cl <- Clip("a",
  source_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24)))
trimmed_range(cl)
```

---

trimmed\_range\_in\_parent

*Range of an item within its parent, trimmed by the parent's source range*

---

**Description**

Range of an item within its parent, trimmed by the parent's source range

**Usage**

```
trimmed_range_in_parent(x)
```

**Arguments**

x                    An item with a parent.

**Value**

A [TimeRange](#).

**Examples**

```
trk <- Track("V1",
  source_range = TimeRange(RationalTime(12, 24), RationalTime(24, 24)))
cl <- Clip("a",
  source_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24)))
append_child(trk, cl)
trimmed_range_in_parent(cl)
```

---

type_version_map	<i>Current OTIO schema versions</i>
------------------	-------------------------------------

---

**Description**

A named integer vector mapping each schema type to its current version, matching OpenTimelineIO 0.18.1.

**Usage**

```
type_version_map()
```

**Value**

A named integer vector.

**Examples**

```
type_version_map()["Clip"]
```

---

validate_with_RcppOTIO	
------------------------	--

*Validate an OTIO object against the real OpenTimelineIO library*

---

**Description**

Serializes `x` to OTIO JSON, parses it with RcppOTIO (the exact libopentimelineio binding), and re-serializes through RcppOTIO. Confirms the emitted JSON is accepted by real OTIO. Requires the optional RcppOTIO package.

**Usage**

```
validate_with_RcppOTIO(x)
```

**Arguments**

`x` An OTIO object (typically a [Timeline](#)).

**Value**

Invisibly, a list with status ("valid" or "unverified") and, when validated, the RcppOTIO-normalized JSON.

**Examples**

```
validate_with_RcppOTIO(Timeline("demo"))
```

---

value	<i>RationalTime value and rate</i>
-------	------------------------------------

---

**Description**

RationalTime value and rate

**Usage**

value(x)

rate(x)

**Arguments**

x                    A RationalTime.

**Value**

Numeric scalar, the time's value or rate.

**Examples**

```
value(RationalTime(180, 30))  
rate(RationalTime(180, 30))
```

---

video_tracks	<i>Video / audio tracks of a timeline</i>
--------------	---

---

**Description**

Video / audio tracks of a timeline

**Usage**

video\_tracks(x)

audio\_tracks(x)

**Arguments**

x                    A [Timeline](#).

**Value**

A list of [Tracks](#) of the matching kind.

**Examples**

```
t1 <- add_track(Timeline("demo"), Track("V1", kind = "Video"))
video_tracks(t1)
audio_tracks(t1)
```

---

visible	<i>Is an item visible?</i>
---------	----------------------------

---

**Description**

A Gap is never visible; a Transition always is; any other item is visible when enabled (matching OTIO).

**Usage**

```
visible(x)
```

**Arguments**

x	An item.
---	----------

**Value**

Logical scalar.

**Examples**

```
visible(Gap(RationalTime(24, 24)))
visible(Clip("a", ExternalReference("a.mp4")))
```

---

visible_range	<i>Visible range of an item (including adjacent transitions)</i>
---------------	--

---

**Description**

Visible range of an item (including adjacent transitions)

**Usage**

```
visible_range(x)
```

**Arguments**

x	An item with a parent.
---	------------------------

**Value**

A [TimeRange](#).

**Examples**

```
trk <- Track("V1")
cl <- Clip("a",
  source_range = TimeRange(RationalTime(0, 24), RationalTime(48, 24)))
append_child(trk, cl)
visible_range(cl)
```

---

write_otiod	<i>Write an OTIOD bundle</i>
-------------	------------------------------

---

**Description**

Writes timeline to <dir>/content.otio and ensures a media/ subdirectory exists. Optionally copies media files into it. Media references inside timeline should already be relative paths (e.g. "media/clip01.mp4"); this writer does not rewrite them.

**Usage**

```
write_otiod(timeline, dir, media = NULL, indent = 2)
```

**Arguments**

timeline	A <a href="#">Timeline</a> .
dir	Bundle directory (created if needed).
media	Optional character vector of media files to copy into <dir>/media/.
indent	Indent width for content.otio (default 2).

**Value**

The path to the written content.otio, invisibly.

**Examples**

```
d <- tempfile("bundle")
write_otiod(Timeline("demo"), d)
dir(d)
unlink(d, recursive = TRUE)
```

# Index

active\_media\_reference\_key, 4  
active\_media\_reference\_key<-  
    (active\_media\_reference\_key), 4  
add\_child, 5, 7, 68  
add\_effect, 5  
add\_track, 6, 63  
almost\_equal, 7  
append\_child, 5, 7, 68  
audio\_tracks (video\_tracks), 75  
available\_range, 8  
available\_range<- (available\_range), 8

children, 8  
clamped, 9  
clear\_children, 10  
Clip, 4, 10, 38, 39, 56, 60  
clone, 11  
color, 11  
color<- (color), 11  
comment, 12  
comment<- (comment), 12  
contains, 13

default\_media\_key, 13  
duration (start\_time), 60

Effect, 6, 14, 15, 36  
effect\_name, 15  
effect\_name<- (effect\_name), 15  
effects, 15  
enabled, 16  
enabled<- (enabled), 16  
end\_frame, 16  
end\_time\_exclusive, 17  
end\_time\_inclusive  
    (end\_time\_exclusive), 17  
extended\_by, 18  
ExternalReference, 18, 60

fill, 19, 36

find\_clips, 20  
flatten\_stack, 20  
frame\_for\_time, 21  
frame\_step (target\_url\_base), 61  
frame\_zero\_padding (target\_url\_base), 61  
FreezeFrame (Effect), 14  
from\_frames, 21  
from\_json\_file, 22  
from\_json\_string, 22, 49  
from\_seconds, 23  
from\_time\_string, 23  
from\_timecode, 24

Gap, 24  
generator\_kind, 25  
generator\_kind<- (generator\_kind), 25  
GeneratorReference, 25, 26, 44  
global\_start\_time, 26  
global\_start\_time<-  
    (global\_start\_time), 26

has\_child, 27  
has\_clips, 28

ImageSequenceReference, 16, 21, 28, 41, 45,  
    61, 62  
in\_offset (transition\_type), 71  
in\_offset<- (transition\_type), 71  
index\_of\_child, 29  
insert, 30  
insert\_child, 31  
intersects, 31  
is\_composition (is\_otio), 33  
is\_effect, 32  
is\_equivalent\_to, 32  
is\_media\_reference (is\_otio), 33  
is\_missing\_reference, 33  
is\_otio, 33  
is\_parent\_of, 34  
is\_rational\_time, 35

- is\_time\_range (is\_rational\_time), 35
- is\_timeline (is\_otio), 33
- is\_unknown\_schema, 35
- Item, 36
- kind, 36
- kind<- (kind), 36
- LinearTimeWarp, 62
- LinearTimeWarp (Effect), 14
- marked\_range, 37
- marked\_range<- (marked\_range), 37
- Marker, 12, 36, 37, 38
- media\_reference, 38
- media\_reference<- (media\_reference), 38
- media\_references, 39
- MediaReference, 39
- metadata, 40
- metadata<- (metadata), 40
- MissingReference (ExternalReference), 18
- name, 41
- name<- (name), 41
- name\_prefix (target\_url\_base), 61
- name\_suffix (target\_url\_base), 61
- number\_of\_images\_in\_sequence, 41
- out\_offset (transition\_type), 71
- out\_offset<- (transition\_type), 71
- overlapping, 42
- overlaps, 42
- overwrite, 43
- parameters, 44
- parameters<- (parameters), 44
- parent, 44
- presentation\_time\_for\_image\_number, 45
- range\_from\_start\_end\_time, 45
- range\_in\_parent, 46
- rate (value), 75
- RationalTime, 19, 21, 24, 26, 27, 30, 45, 47, 50–52, 56–58, 63, 64, 70–72
- read\_otiod, 47
- register\_downgrade\_function, 48
- register\_upgrade\_function, 49
- remove, 49
- remove\_child, 7, 50
- rescaled\_to, 51
- ripple, 51
- roll, 52
- schema\_name, 53
- schema\_version, 53
- SerializableCollection, 54
- set\_child, 54
- set\_children, 55
- set\_children<- (set\_children), 55
- set\_media\_references, 56
- slice, 56
- slide, 57
- slip, 58
- source\_range, 58
- source\_range<- (source\_range), 58
- Stack, 20, 59, 63, 70
- start\_frame (target\_url\_base), 61
- start\_time, 60
- target\_url, 60
- target\_url<- (target\_url), 60
- target\_url\_base, 61
- target\_url\_base<- (target\_url\_base), 61
- target\_url\_for\_image\_number, 62
- time\_scalar, 62
- time\_scalar<- (time\_scalar), 62
- TimeEffect (Effect), 14
- Timeline, 6, 23, 26, 47, 63, 65, 66, 70, 74, 75, 77
- TimeRange, 8, 10, 18, 26, 29, 36–38, 40, 43, 46, 58, 59, 64, 68, 69, 73, 77
- TimeTransform, 64
- to\_frames, 65
- to\_json\_file, 65
- to\_json\_string, 65, 66
- to\_seconds, 67
- to\_time\_string, 67
- to\_timecode, 68
- Track, 6, 19, 20, 30, 37, 43, 50, 56, 68, 69, 75
- track\_trimmed\_to\_range, 69
- tracks, 70
- tracks<- (tracks), 70
- Transition, 70, 71
- transition\_type, 71
- transition\_type<- (transition\_type), 71
- trim, 72
- trimmed\_range, 72
- trimmed\_range\_in\_parent, 73
- type\_version\_map, 74

validate\_with\_RcppOTIO, [66](#), [74](#)  
value, [75](#)  
video\_tracks, [75](#)  
visible, [76](#)  
visible\_range, [76](#)  
  
write\_otiod, [77](#)