Liquidsoap 2

A preview!

Romain Beauxis, Jan. 17, 2021

Timeline

Recent Changes

Timeline

Recent Changes

- July 24, 2019: First 1.5.0 changelog, internal video format switched to YUV420p

Timeline

Recent Changes

- July 24, 2019: First 1.5.0 changelog, internal video format switched to YUV420p

1.5.0

Timeline

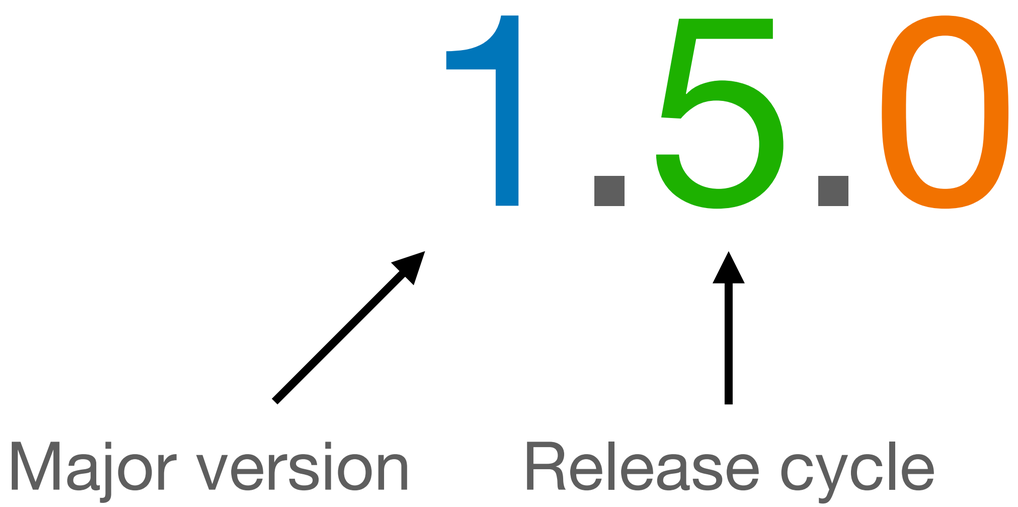
Recent Changes

- July 24, 2019: First 1.5.0 changelog, internal video format switched to YUV420p

Timeline

Recent Changes

- July 24, 2019: First 1.5.0 changelog, internal video format switched to YUV420p



Timeline

Recent Changes

- July 24, 2019: First 1.5.0 changelog, internal video format switched to YUV420p

Timeline

Recent Changes

- July 24, 2019: First 1.5.0 changelog, internal video format switched to YUV420p
- Sept. 29, 2019: Release 1.4.0

Timeline

Recent Changes

- July 24, 2019: First 1.5.0 changelog, internal video format switched to YUV420p
- Sept. 29, 2019: Release 1.4.0
- Oct. 31, 2019: Native implementation of request.queue

Timeline

Recent Changes

- July 24, 2019: First 1.5.0 changelog, internal video format switched to YUV420p
- Sept. 29, 2019: Release 1.4.0
- Oct. 31, 2019: Native implementation of request.queue
- Feb. 1, 2020: FOSDEM presentation

Timeline

Recent Changes

- July 24, 2019: First 1.5.0 changelog, internal video format switched to YUV420p
- Sept. 29, 2019: Release 1.4.0
- Oct. 31, 2019: Native implementation of request.queue
- Feb. 1, 2020: FOSDEM presentation
- Feb. 10, 2020: Add support for video encoding/decoding using FFmpeg

Timeline

Recent Changes

- July 24, 2019: First 1.5.0 changelog, internal video format switched to YUV420p
- Sept. 29, 2019: Release 1.4.0
- Oct. 31, 2019: Native implementation of request.queue
- Feb. 1, 2020: FOSDEM presentation
- Feb. 10, 2020: Add support for video encoding/decoding using FFmpeg
- May 25, 2020: Compute source kind at runtime, get rid of runtime types

Timeline

Recent Changes

Timeline

Recent Changes

- June 7, 2020: Add support for modules

Timeline

Recent Changes

- June 7, 2020: Add support for modules
- June 18, 2020: Add support for errors and nullable types

Timeline

Recent Changes

- June 7, 2020: Add support for modules
- June 18, 2020: Add support for errors and nullable types
- June 20, 2020: Add support for for and while loops

Timeline

Recent Changes

- June 7, 2020: Add support for modules
- June 18, 2020: Add support for errors and nullable types
- June 20, 2020: Add support for for and while loops
- June 29, 2020: Add support for `x ? y : z` and list and array splats

Timeline

Recent Changes

- June 7, 2020: Add support for modules
- June 18, 2020: Add support for errors and nullable types
- June 20, 2020: Add support for for and while loops
- June 29, 2020: Add support for `x ? y : z` and list and array splats
- July 3, 2020: Add support for extensible frame content

Timeline

Recent Changes

- June 7, 2020: Add support for modules
- June 18, 2020: Add support for errors and nullable types
- June 20, 2020: Add support for for and while loops
- June 29, 2020: Add support for `x ? y : z` and list and array splats
- July 3, 2020: Add support for extensible frame content
- July 30, 2020: Add support for ffmpeg copy encoder/decoder

Timeline

Recent Changes

- June 7, 2020: Add support for modules
- June 18, 2020: Add support for errors and nullable types
- June 20, 2020: Add support for for and while loops
- June 29, 2020: Add support for `x ? y : z` and list and array splats
- July 3, 2020: Add support for extensible frame content
- July 30, 2020: Add support for ffmpeg copy encoder/decoder
- August 28, 2020: Add support for ffmpeg raw encoder/decoder

Timeline

Recent Changes

Timeline

Recent Changes

- Sept. 10, 2020: Switch to GitHub actions, build docker images, Debian packages and windows binary as part of the CI workflow

Timeline

Recent Changes

- Sept. 10, 2020: Switch to GitHub actions, build docker images, Debian packages and windows binary as part of the CI workflow
- Sept. 16, 2020: 1.4.3 bug fix release.

Timeline

Recent Changes

- Sept. 10, 2020: Switch to GitHub actions, build docker images, Debian packages and windows binary as part of the CI workflow
- Sept. 16, 2020: 1.4.3 bug fix release.
- Oct. 20, 2020: Add support for SRT listener/caller for both input and output

Timeline

Recent Changes

- Sept. 10, 2020: Switch to GitHub actions, build docker images, Debian packages and windows binary as part of the CI workflow
- Sept. 16, 2020: 1.4.3 bug fix release.
- Oct. 20, 2020: Add support for SRT listener/caller for both input and output
- Nov. 15, 2020: Add video, mp4 support to HLS output.

Timeline

Recent Changes

- Sept. 10, 2020: Switch to GitHub actions, build docker images, Debian packages and windows binary as part of the CI workflow
- Sept. 16, 2020: 1.4.3 bug fix release.
- Oct. 20, 2020: Add support for SRT listener/caller for both input and output
- Nov. 15, 2020: Add video, mp4 support to HLS output.
- Dec. 12, 2020: FFmpeg inline conversions for raw frames

Timeline

Recent Changes

- Sept. 10, 2020: Switch to GitHub actions, build docker images, Debian packages and windows binary as part of the CI workflow
- Sept. 16, 2020: 1.4.3 bug fix release.
- Oct. 20, 2020: Add support for SRT listener/caller for both input and output
- Nov. 15, 2020: Add video, mp4 support to HLS output.
- Dec. 12, 2020: FFmpeg inline conversions for raw frames
- Dec. 25, 2020: FFmpeg inline conversion for encoded frames

Language Changes

More expressivity

- Module and records

Language Changes

More expressivity

output.icecast

- Module and records

Language Changes

More expressivity

- Module and records

output.icecast

Language Changes

More expressivity

- Module and records

output.icecast

```
.{
   dummy : 'a.'b.'c.(?id : string, ?fallible : bool,
                     ?on_start : (() -> unit), ?on_stop : (() -> unit),
                     ?start : bool, source(audio='a, video='b, midi='c)) ->
                     active_source(audio='a, video='b, midi='c)

   url : 'a.(?id : string, ?fallible : bool, ?on_start : (() -> unit),
             ?on_stop : (() -> unit), ?start : bool, url : string,
             format('a), source('a)) -> active_source(‘a)
…
}
```

Language Changes
More expressivity

- Module and records

```
http.get

-  : (?headers : [string * string],
       ?http_version : string, ?redirect : bool,
       ?timeout : float, string) ->
  string.{
     headers : [string * string],
     status_message : string,
     status_code : int,
     protocol_version : string
  }
```

Language Changes
More expressivity

- Module and records

```
http.get

-  : (?headers : [string * string],
       ?http_version : string, ?redirect : bool,
       ?timeout : float, string) ->
  string.{
     headers : [string * string],
     status_message : string,
     status_code : int,
     protocol_version : string
  }
```

```
"<html>...</html>".{
  headers = [
    ("connection", "close"),
    ("content-length", "16977"),
    ...
  ],
  status_message = "OK",
  status_code = 200,
  protocol_version = "HTTP/1.1"
}
```

Language Changes
More expressivity

- Module and records

```
http.get

-   : (?headers : [string * string],
       ?http_version : string, ?redirect : bool,
       ?timeout : float, string) ->
  string.{
     headers : [string * string],
     status_message : string,
     status_code : int,
     protocol_version : string
  }
```

```
x = https.get("https://liquidsoap.info")

x.status_code
-   : int = 200

x.status_message
-   : string = "OK"

"html: #{x}"
- : string = "html: <HTML>…</HTML>\r\n"
```

Language Changes

More expressivity

- Module and records

```
source(audio=?A, video=?B, midi=none)
  .{
    time : () -> float,
    shutdown : () -> unit,
    fallible : () -> bool,
    skip : () -> unit,
    seek : (float) -> float,
    is_up : () -> bool,
    remaining : () -> float,
    on_track : ((([string * string]) -> unit)) -> unit,
    on_shutdown : ((() -> unit)) -> unit,
    on_metadata : ((([string * string]) -> unit)) -> unit,
    is_ready : () -> bool,
    id : (() -> string)
  }
```

Language Changes

More expressivity

- Module and records

```
s = single("/path/to/file.mp3")

s.skip()
```

```
source(audio=?A, video=?B, midi=none)
  .{
    time : () -> float,
    shutdown : () -> unit,
    fallible : () -> bool,
    skip : () -> unit,
    seek : (float) -> float,
    is_up : () -> bool,
    remaining : () -> float,
    on_track : ((([string * string]) -> unit)) -> unit,
    on_shutdown : ((() -> unit)) -> unit,
    on_metadata : ((([string * string]) -> unit)) -> unit,
    is_ready : () -> bool,
    id : (() -> string)
  }
```

Language Changes

More expressivity

- Module and records

```
s = single("/path/to/file.mp3")

s.skip()


s = insert_metadata(s)

s.insert_metadata(
  [("title", "Awesome Track")
])
```

```
source(audio=?A, video=?B, midi=none)
  .{
    time : () -> float,
    shutdown : () -> unit,
    fallible : () -> bool,
    skip : () -> unit,
    seek : (float) -> float,
    is_up : () -> bool,
    remaining : () -> float,
    on_track : ((([string * string]) -> unit)) -> unit,
    on_shutdown : ((() -> unit)) -> unit,
    on_metadata : ((([string * string]) -> unit)) -> unit,
    is_ready : () -> bool,
    id : (() -> string)
  }
```

Language Changes

More expressivity

- Module and records

libs/deprecations.liq:

```
server.register(
    "stop",
    namespace=s.id(),
    description="…",
    fun (_) -> begin
      s.stop()
     "Done"
    end)
```

```
source(audio=?A, video=?B, midi=none)
  .{
    time : () -> float,
    shutdown : () -> unit,
    fallible : () -> bool,
    skip : () -> unit,
    seek : (float) -> float,
    is_up : () -> bool,
    remaining : () -> float,
    on_track : ((([string * string]) -> unit)) -> unit,
    on_shutdown : ((() -> unit)) -> unit,
    on_metadata : ((([string * string]) -> unit)) -> unit,
    is_ready : () -> bool,
    id : (() -> string)
  }
```

Language Changes
More expressivity

- Nullable type

```
x = null()

y = x ?? "foo"
y : string = "foo"

x = null("bla")

y = x ?? "foo"
y : string = "bla"
```

Language Changes

More expressivity

- Nullable type

```
x = null()

y = x ?? "foo"
y : string = "foo"

x = null("bla")

y = x ?? "foo"
y : string = "bla"
```

```
def f(~opt=null(), x) =
  y = opt ?? "no arg"
  print(y)
  print(x)
end

f : (?opt : string?, 'a) -> unit = <fun>
```

Language Changes

More expressivity

- Nullable type

```
x = null()

y = x ?? "foo"
y : string = "foo"

x = null("bla")

y = x ?? "foo"
y : string = "bla"
```

```
def f(~opt=null(), x) =
  y = opt ?? "no arg"
  print(y)
  print(x)
end

f : (?opt : string?, 'a) -> unit = <fun>

f("foo")
no arg
foo
```

Language Changes

More expressivity

- Nullable type

```
x = null()

y = x ?? "foo"
y : string = "foo"

x = null("bla")

y = x ?? "foo"
y : string = "bla"
```

```
def f(~opt=null(), x) =
  y = opt ?? "no arg"
  print(y)
  print(x)
end

f : (?opt : string?, 'a) -> unit = <fun>

f("foo")
no arg
foo

f(opt="gni", "foo")
gni
foo
```

Language Changes
More expressivity

- Errors

```
x = try
  …
catch err do
  …
end
```

Language Changes
More expressivity

- Errors

```
x = try
  …
catch err in […] do
  …
end
```

Language Changes
More expressivity

- Errors

```
x = try
  …
catch err in […] do
  …
end
```

```
err = error.register("foo")
err : error = error(kind="foo",message=none)

error.message(err)
- : string? = null
error.kind(err)
- : string = "foo"
```

Language Changes
More expressivity

- Errors

```
x = try
  …
catch err do
  msg = error.message(err) ?? "msg"
  …
end
```

```
err = error.register("foo")
err : error = error(kind="foo",message=none)

error.message(err)
- : string? = null
error.kind(err)
- : string = "foo"
```

Language Changes
More expressivity

- Splats

```
let [x, _, z, ...t] = [1,2,3,4]
x : int = 1
z : int = 3
t : [int] = [4]

x = [1, ...[2, 3, 4], 5, ...[6, 7]]
x : [int] = [1, 2, 3, 4, 5, 6, 7]
```

Language Changes
More expressivity

- Splats

```
let [x, _, z, ...t] = [1,2,3,4]
x : int = 1
z : int = 3
t : [int] = [4]

x = [1, ...[2, 3, 4], 5, ...[6, 7]]
x : [int] = [1, 2, 3, 4, 5, 6, 7]
```

```
let (x, _, z) = (1,2,3)
(x, _, z) : int * int * int = (1, 2, 3)
```

FFmpeg integration

Anything is possible!

FFmpeg integration

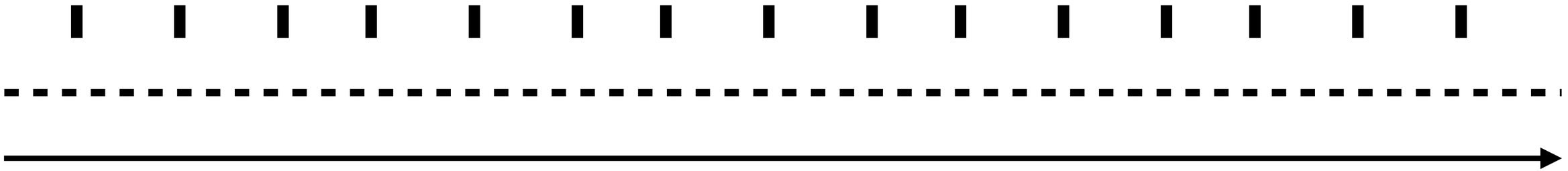
Anything is possible!

FFmpeg integration

Anything is possible!

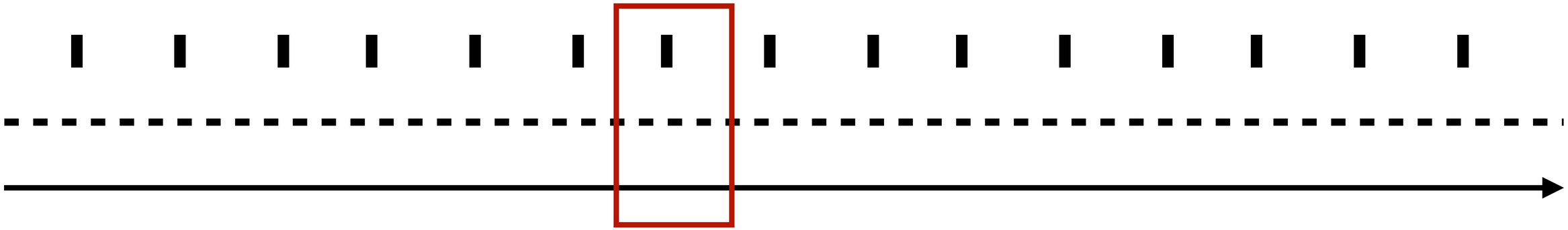
FFmpeg integration

Anything is possible!



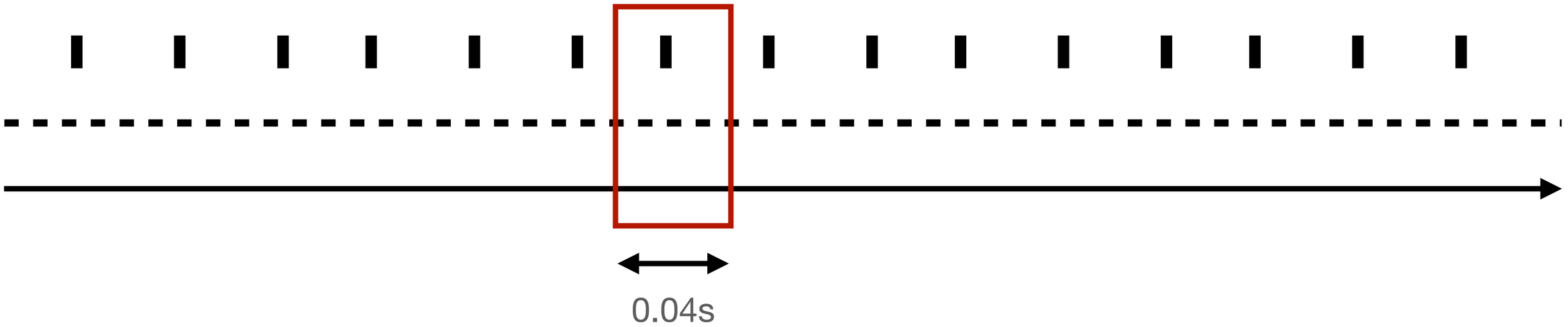
FFmpeg integration

Anything is possible!



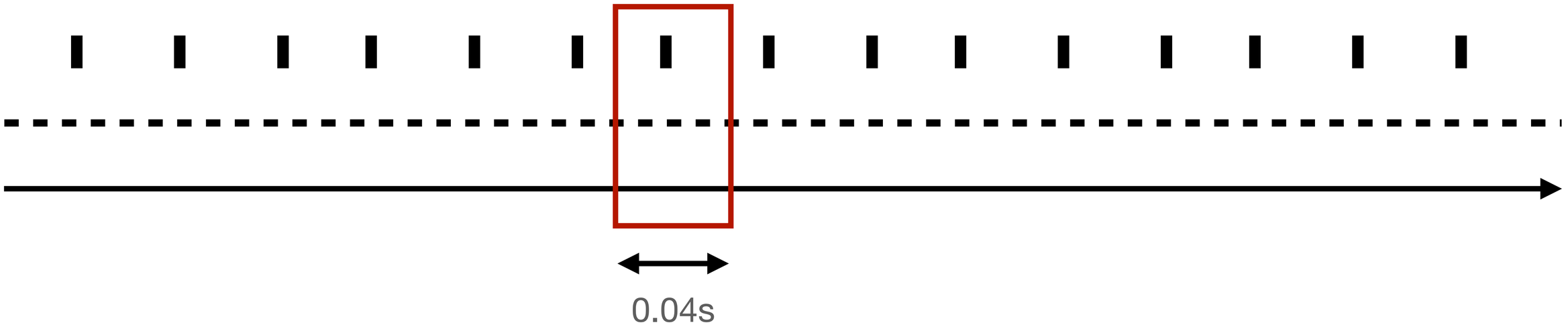
FFmpeg integration

Anything is possible!



FFmpeg integration

Anything is possible!



[frame:3] Frame size must be a multiple of 1764 ticks = 1764 audio samples = 1 video samples.
[frame:3] Targetting 'frame.duration': 0.04s = 1764 audio samples = 1764 ticks.
[frame:3] Frames last 0.04s = 1764 audio samples = 1 video samples = 1764 ticks.

FFmpeg integration

Anything is possible!

```
single("/path/to/file.mp3")
```

FFmpeg integration

Anything is possible!

single("/path/to/file.mp3")

playlist(“/path/to/directory/")

FFmpeg integration

Anything is possible!

single("/path/to/file.mp3")

playlist(“/path/to/directory/")

request.queue(...)

FFmpeg integration

Anything is possible!

single("/path/to/file.mp3")

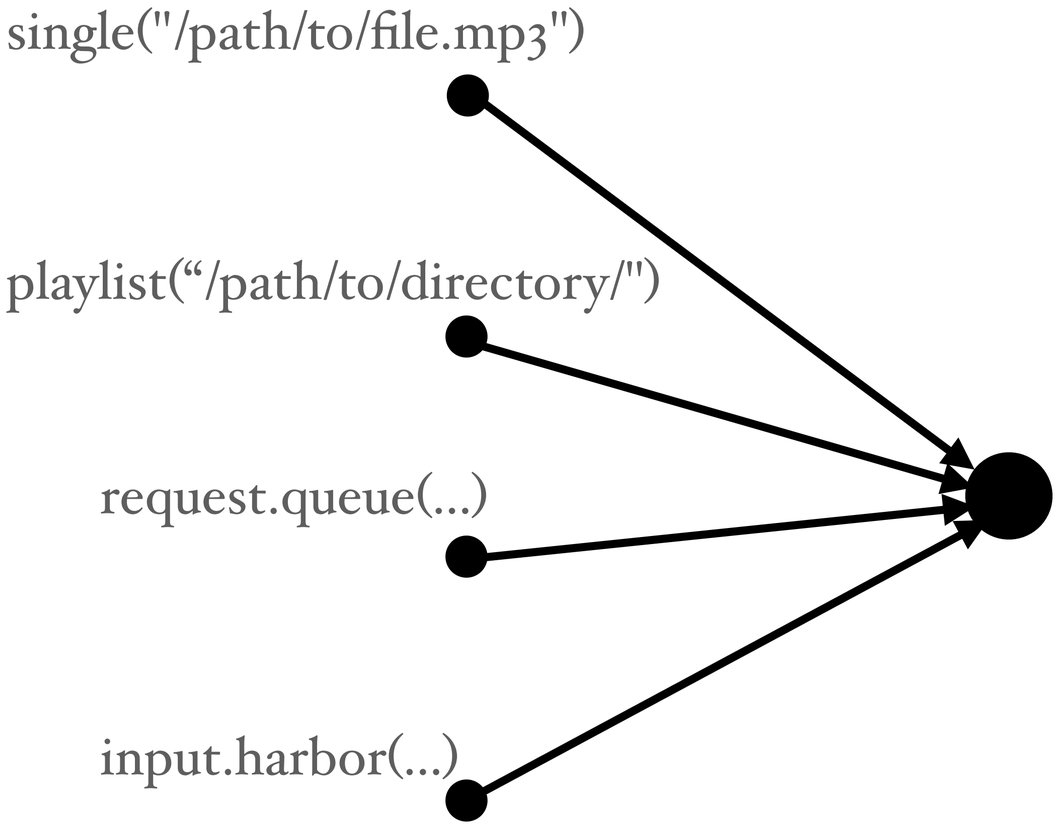
playlist(“/path/to/directory/")

request.queue(...)

input.harbor(...)

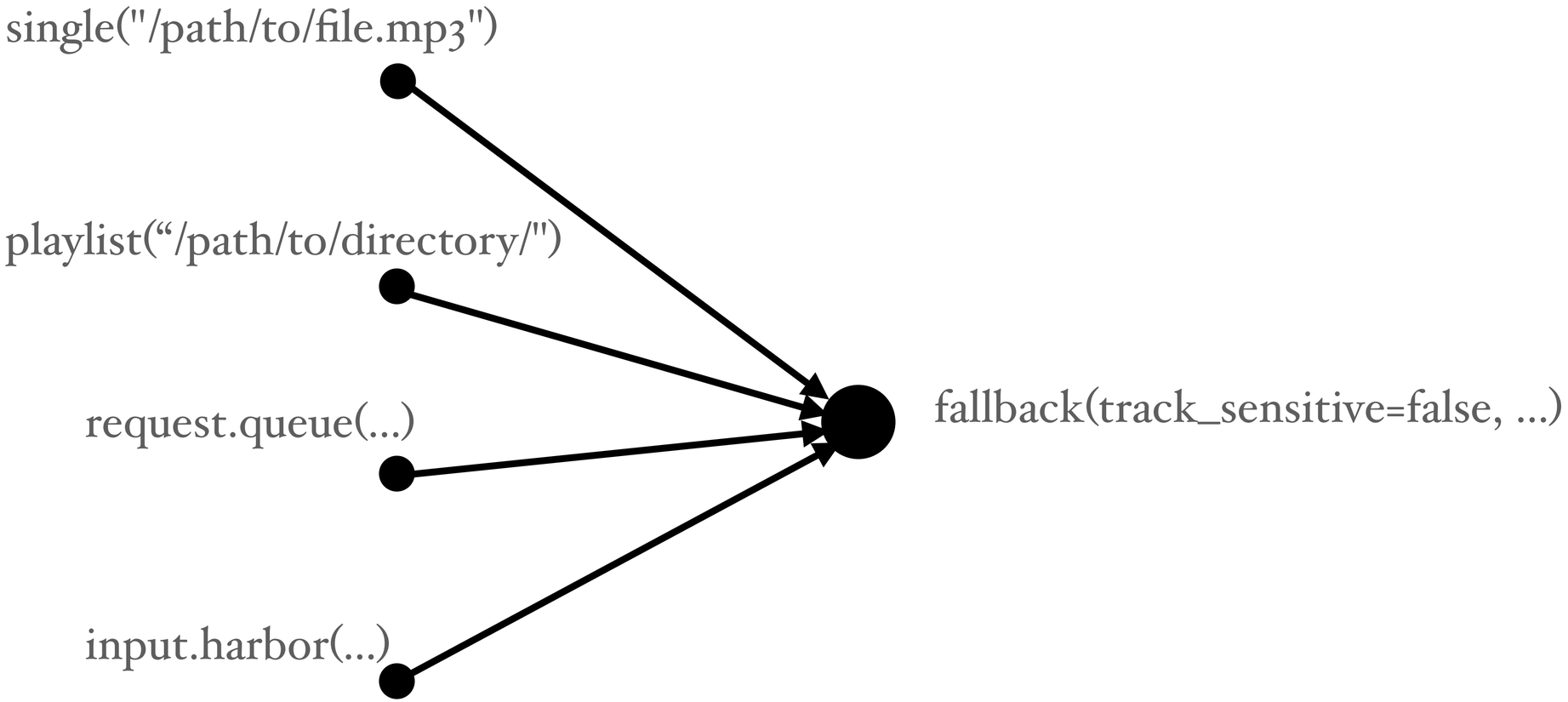
FFmpeg integration

Anything is possible!



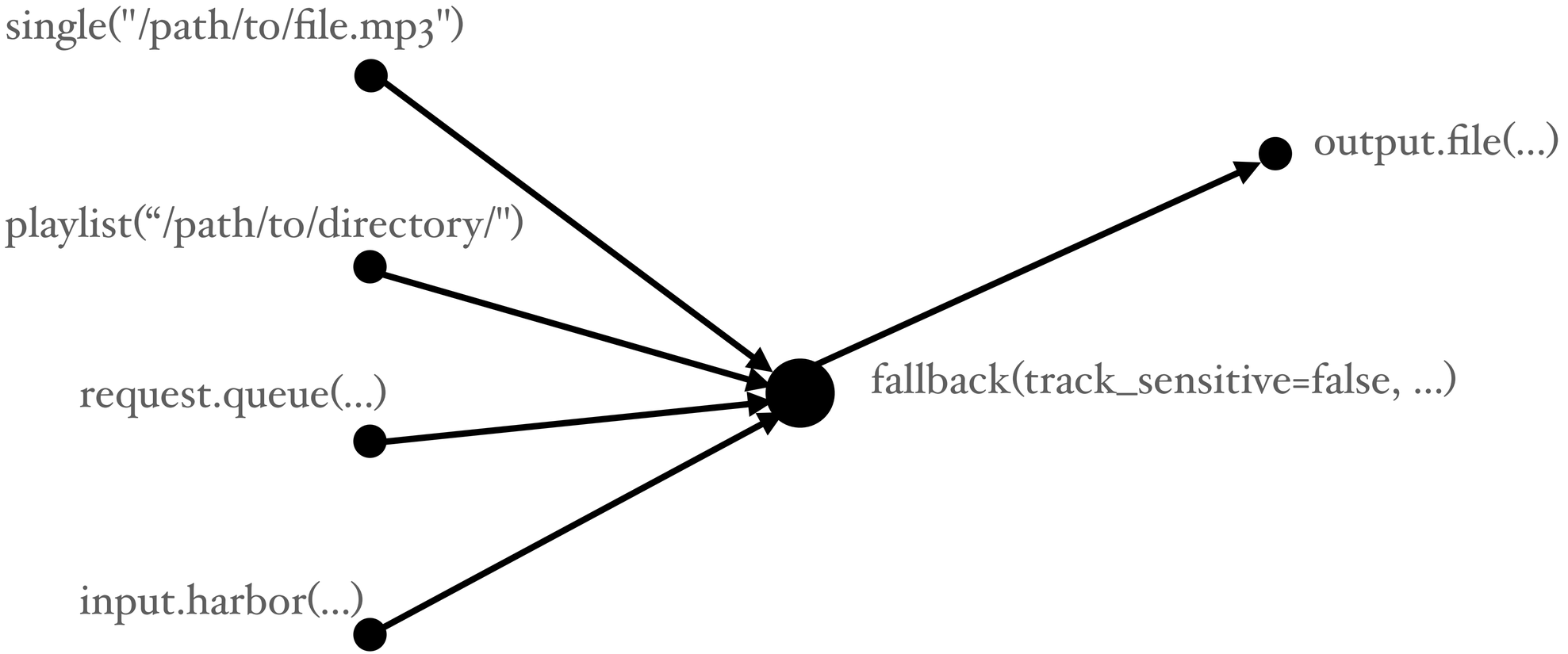
FFmpeg integration

Anything is possible!



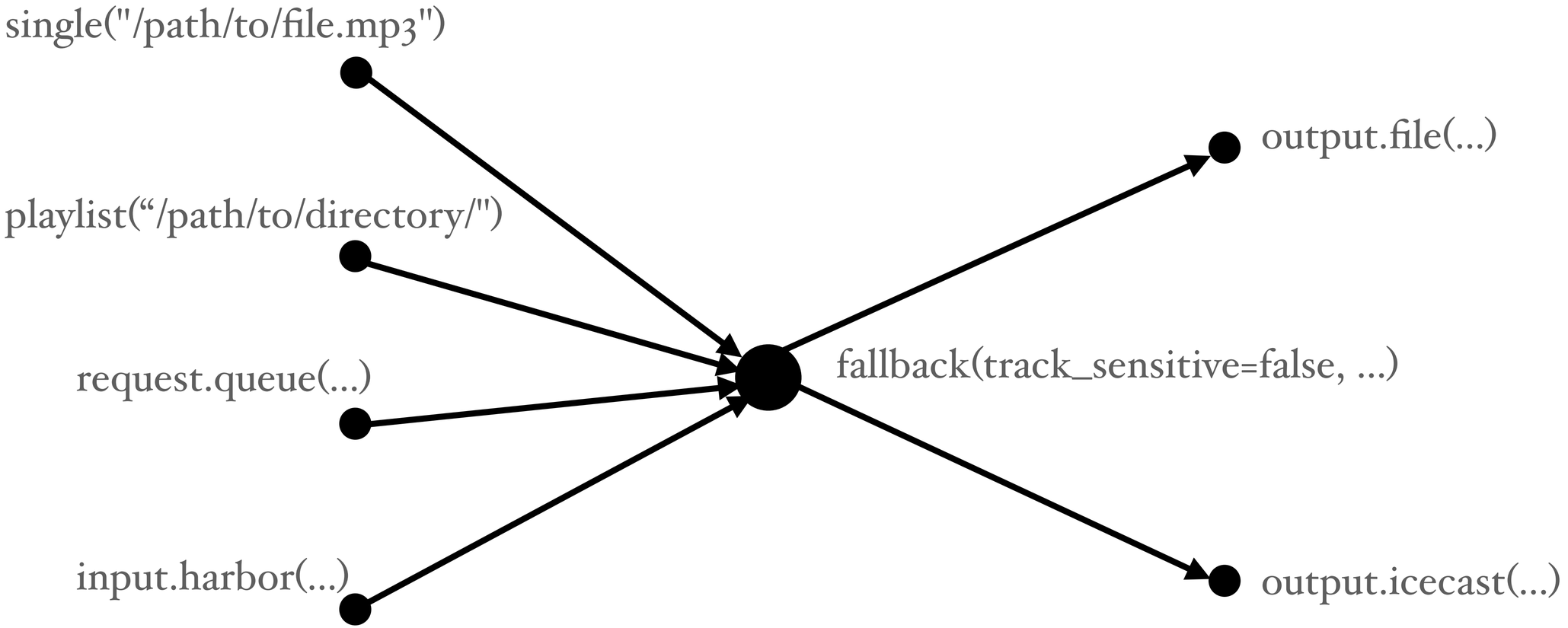
FFmpeg integration

Anything is possible!



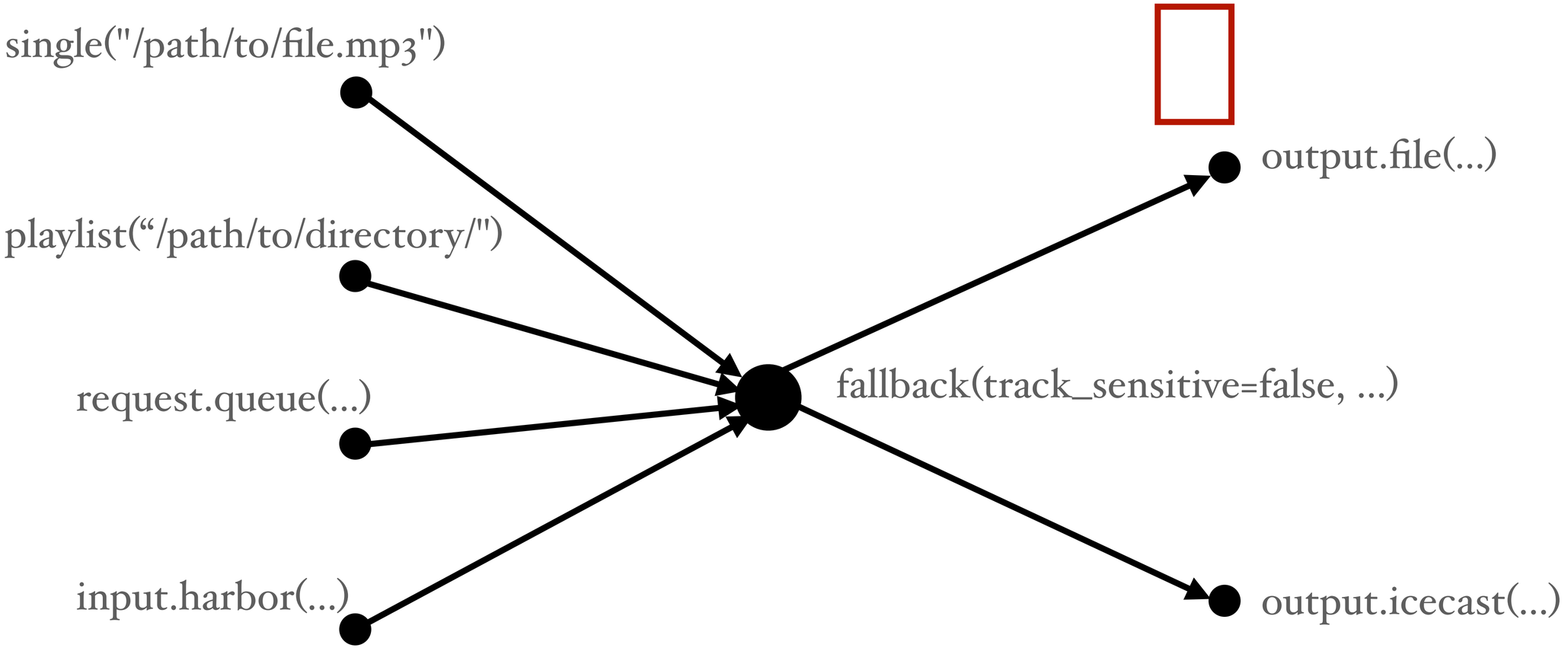
FFmpeg integration

Anything is possible!



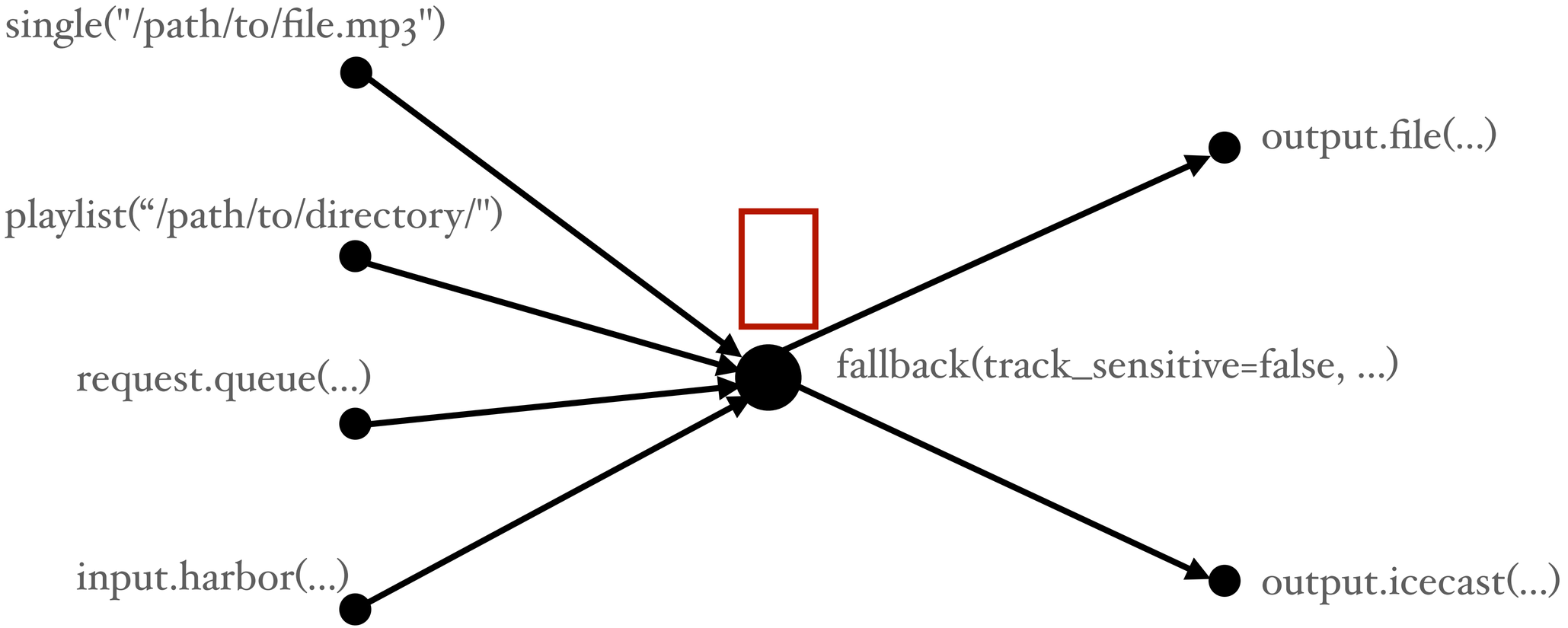
FFmpeg integration

Anything is possible!



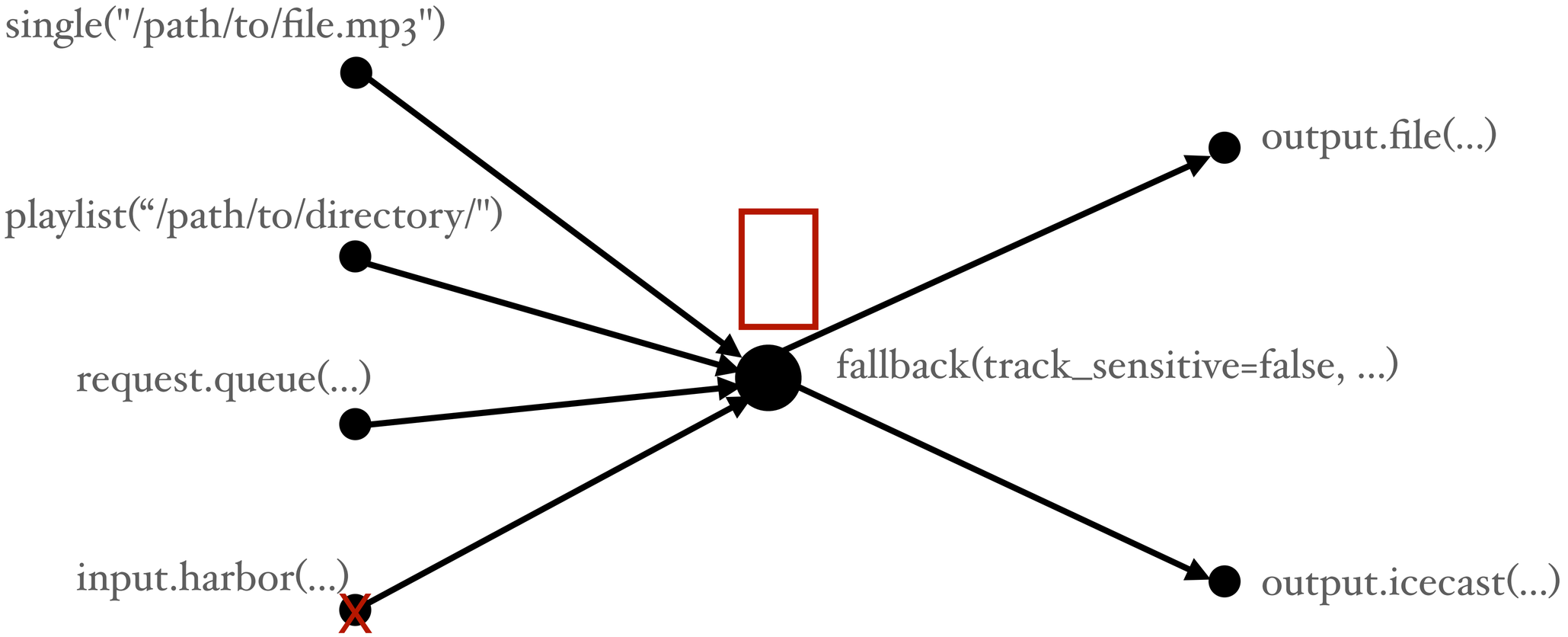
FFmpeg integration

Anything is possible!



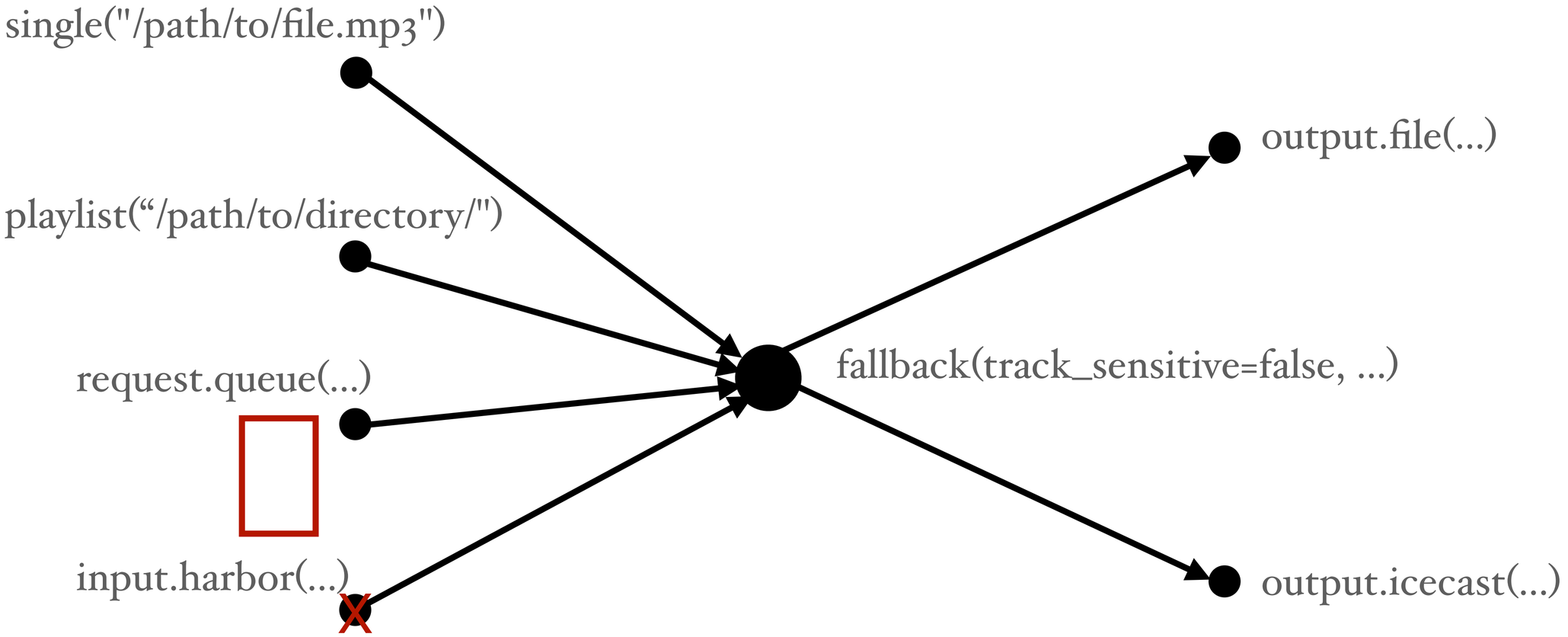
FFmpeg integration

Anything is possible!



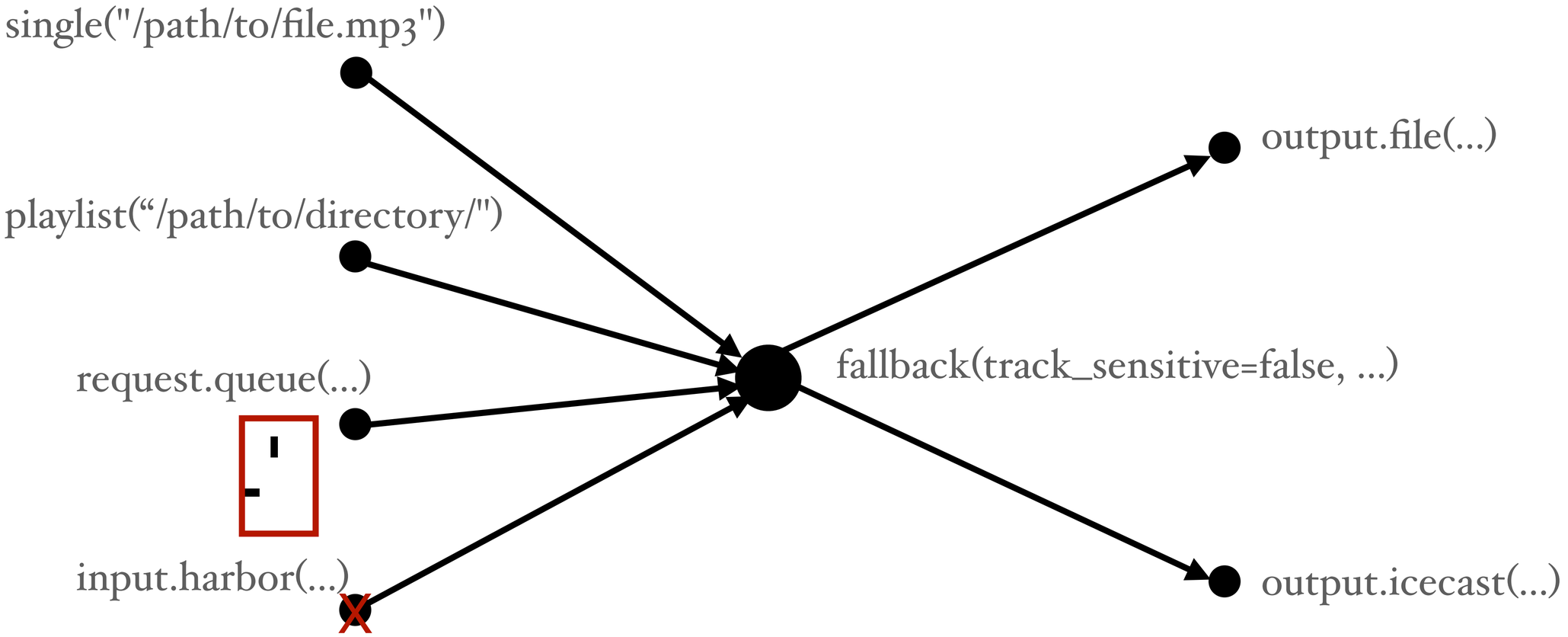
FFmpeg integration

Anything is possible!



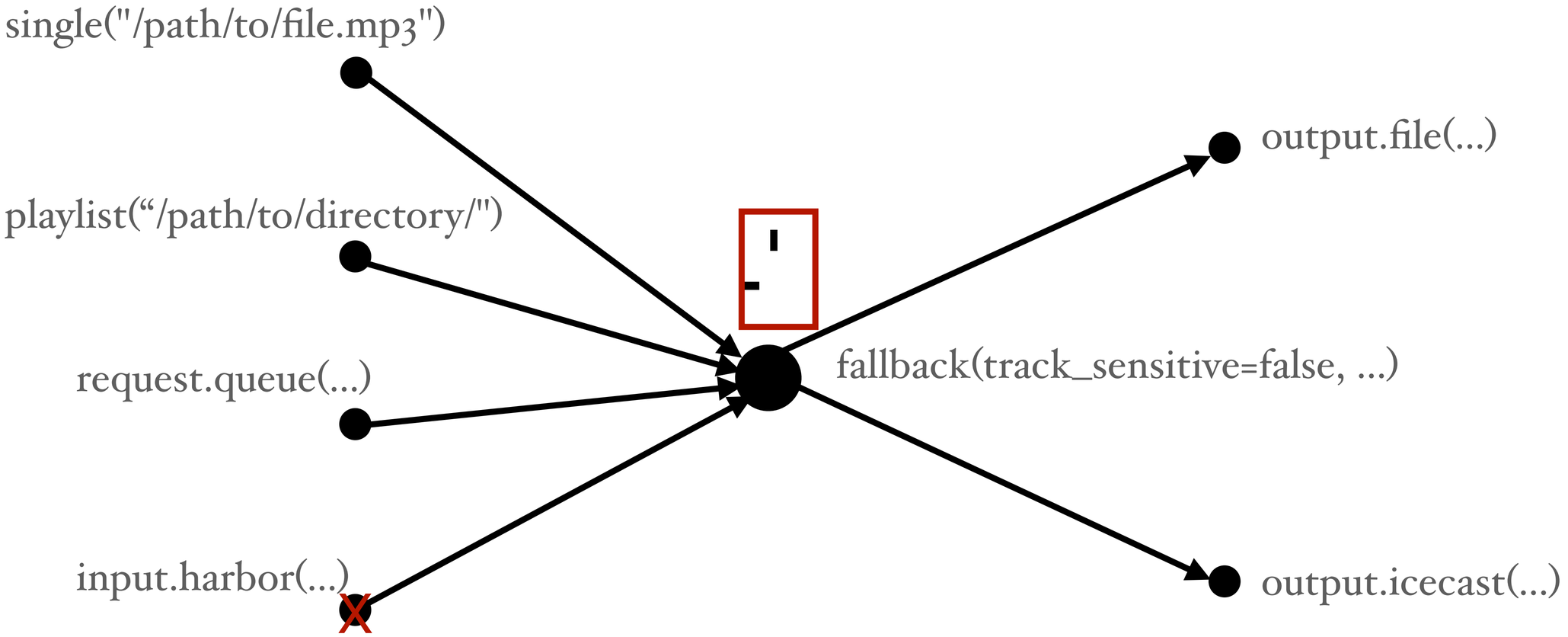
FFmpeg integration

Anything is possible!



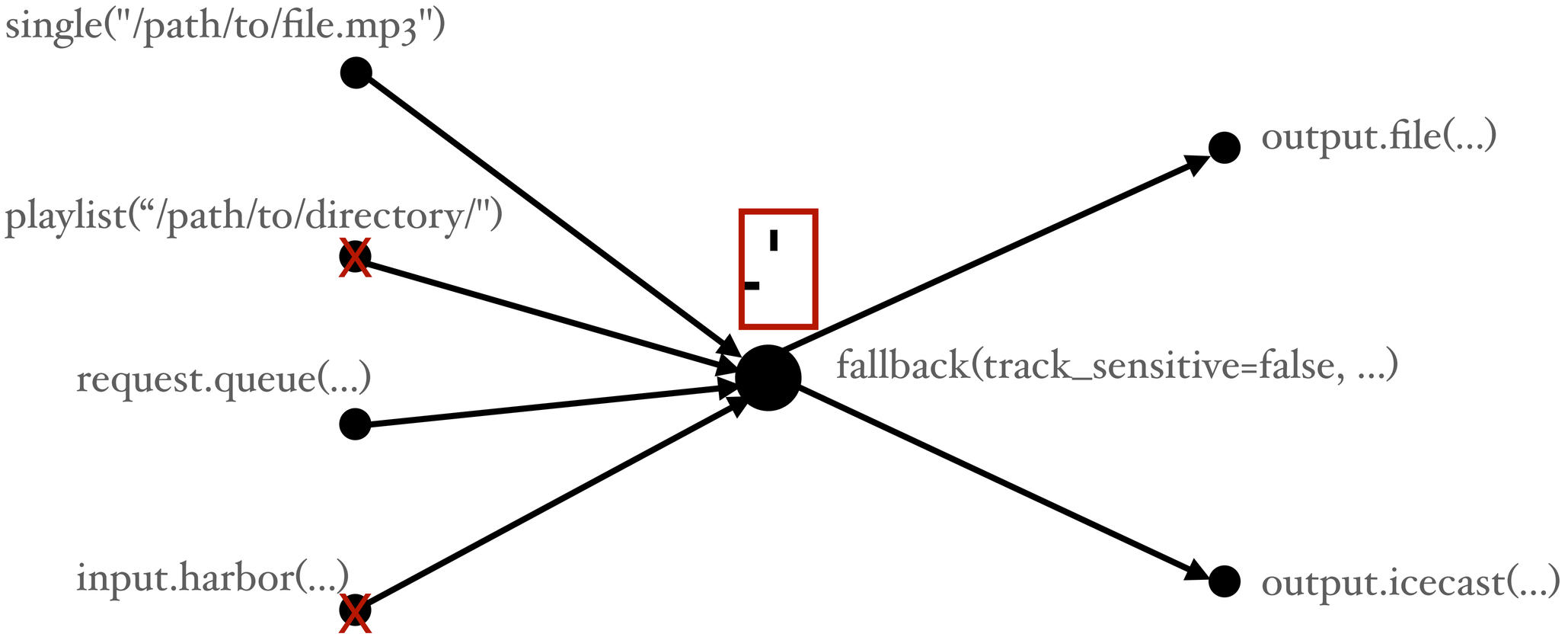
FFmpeg integration

Anything is possible!



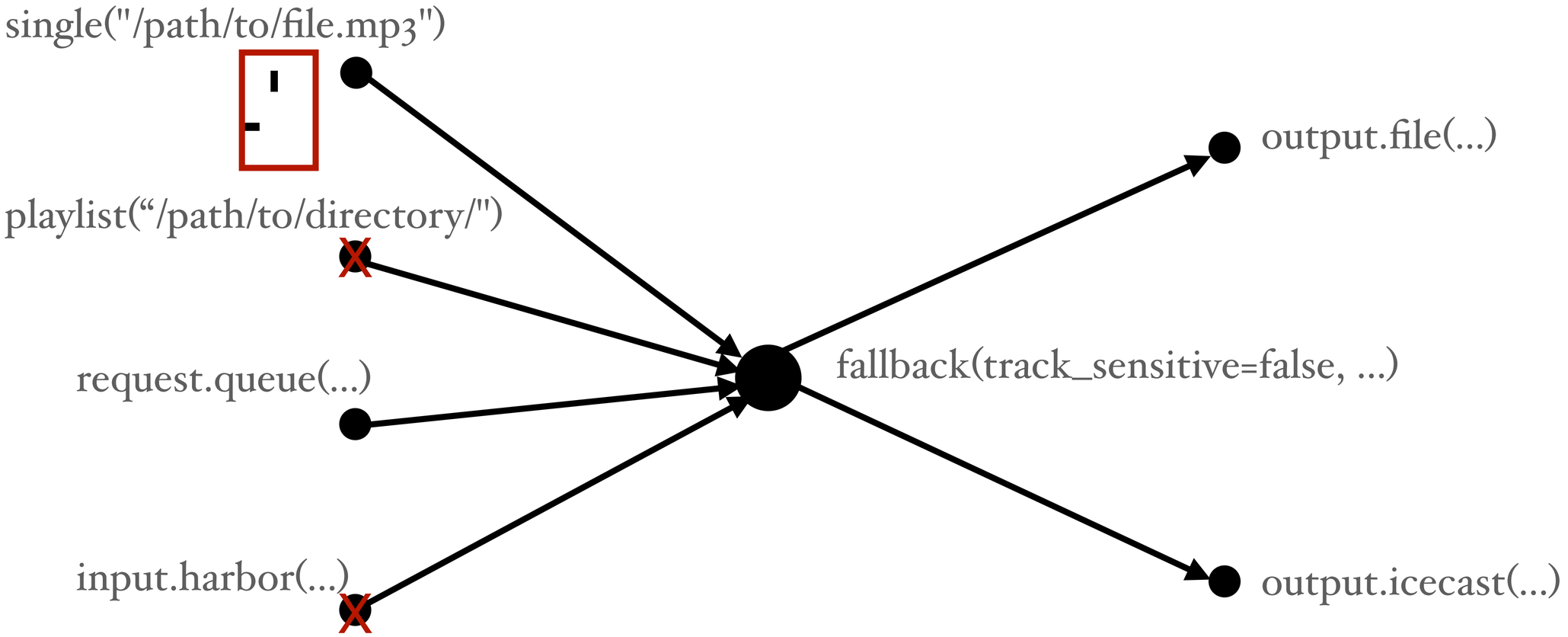
FFmpeg integration

Anything is possible!



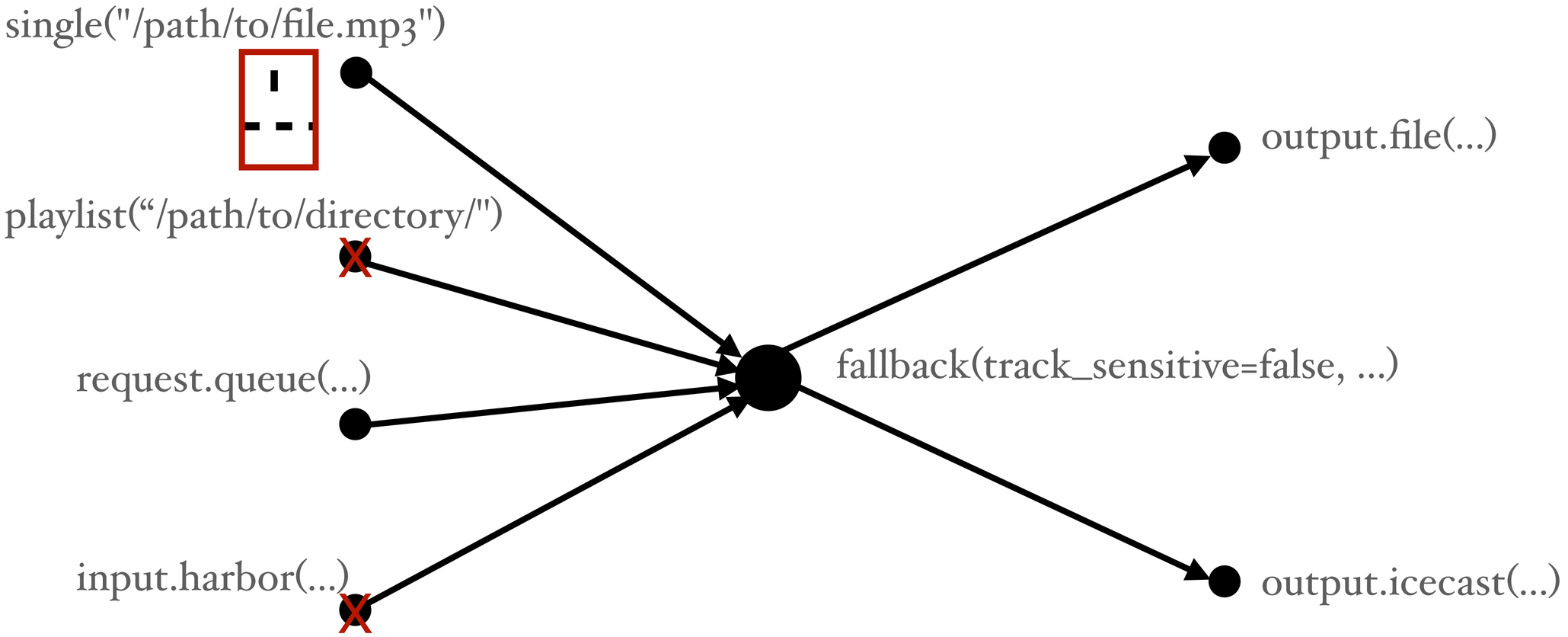
FFmpeg integration

Anything is possible!



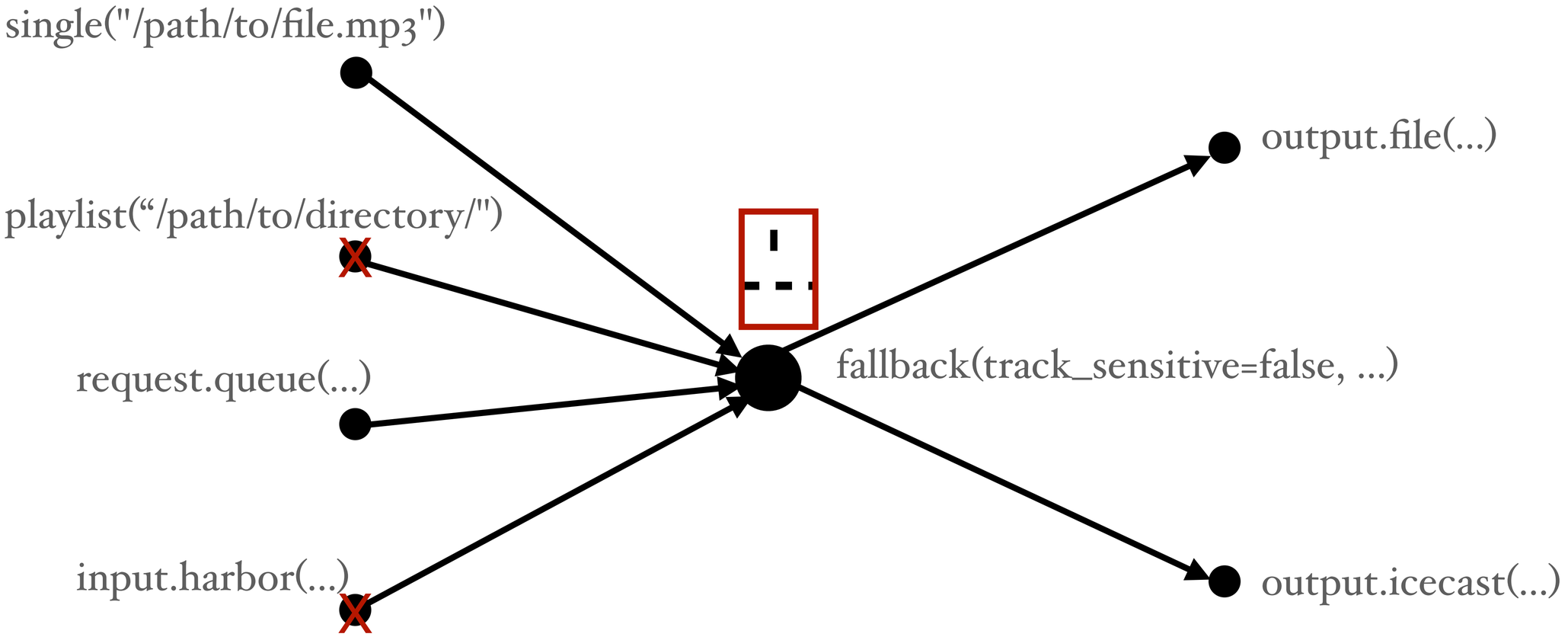
FFmpeg integration

Anything is possible!



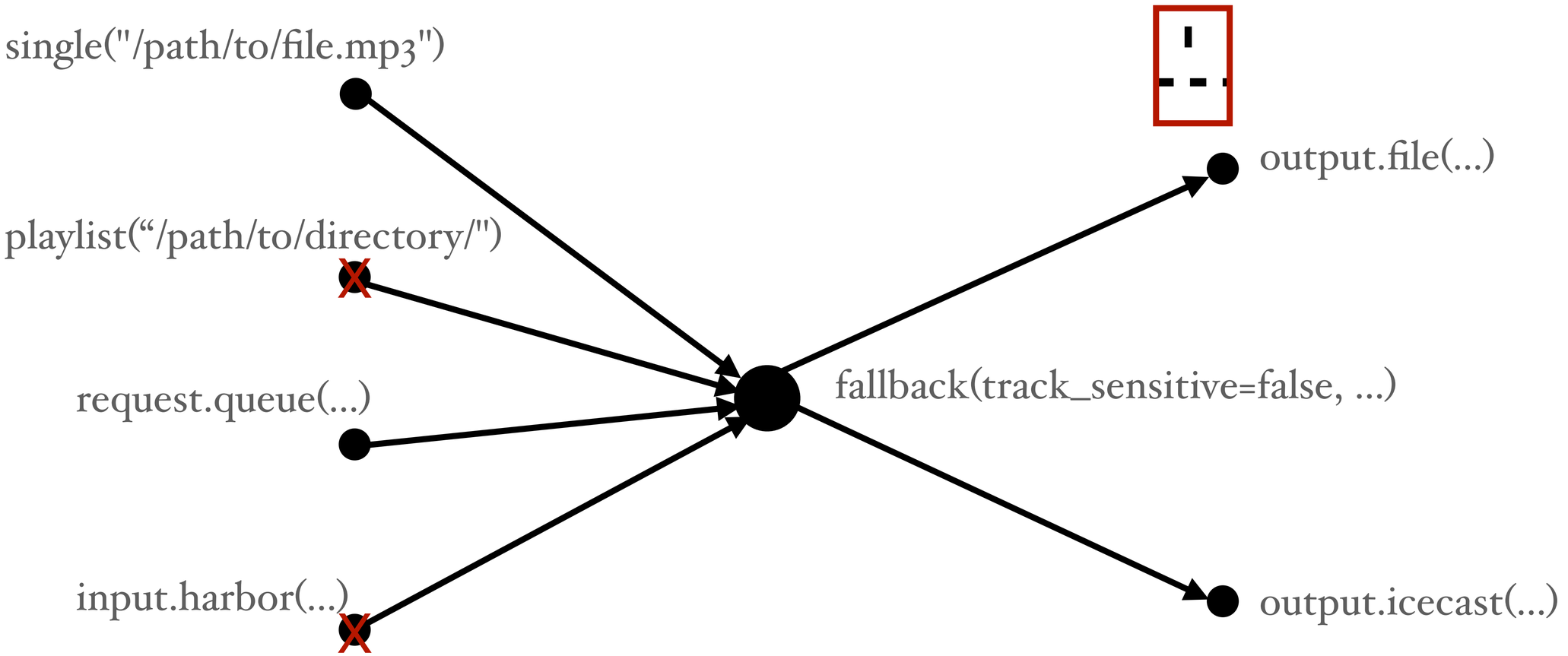
FFmpeg integration

Anything is possible!



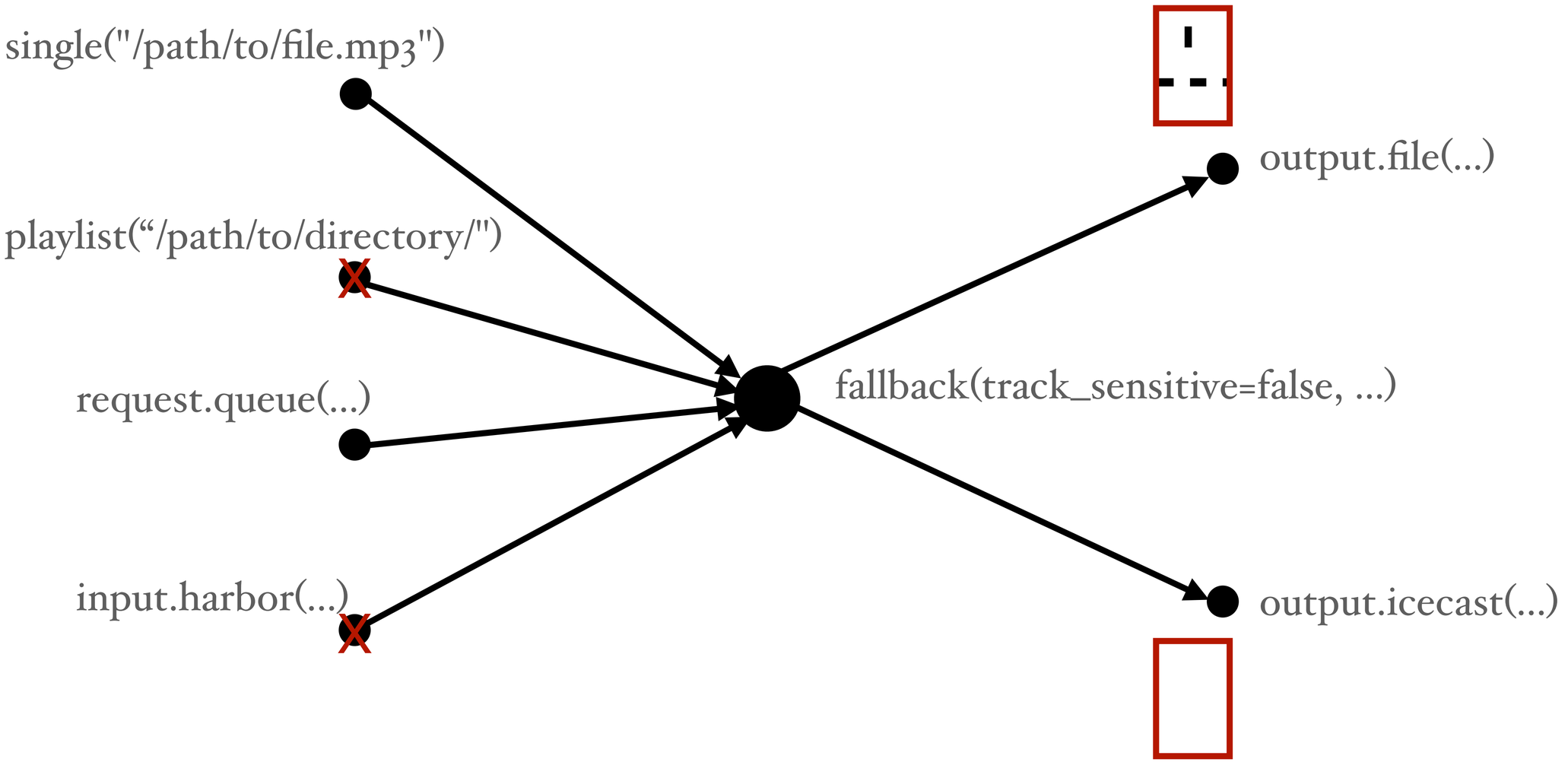
FFmpeg integration

Anything is possible!



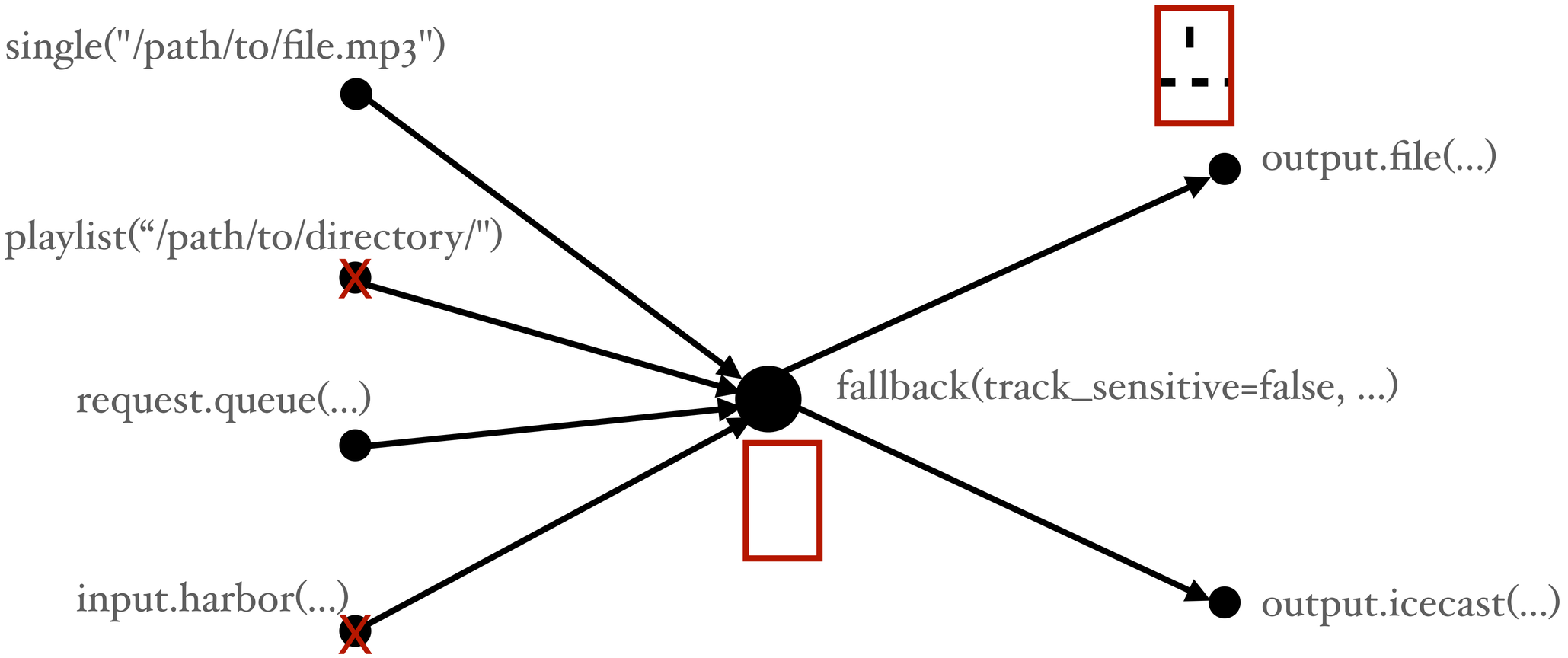
FFmpeg integration

Anything is possible!



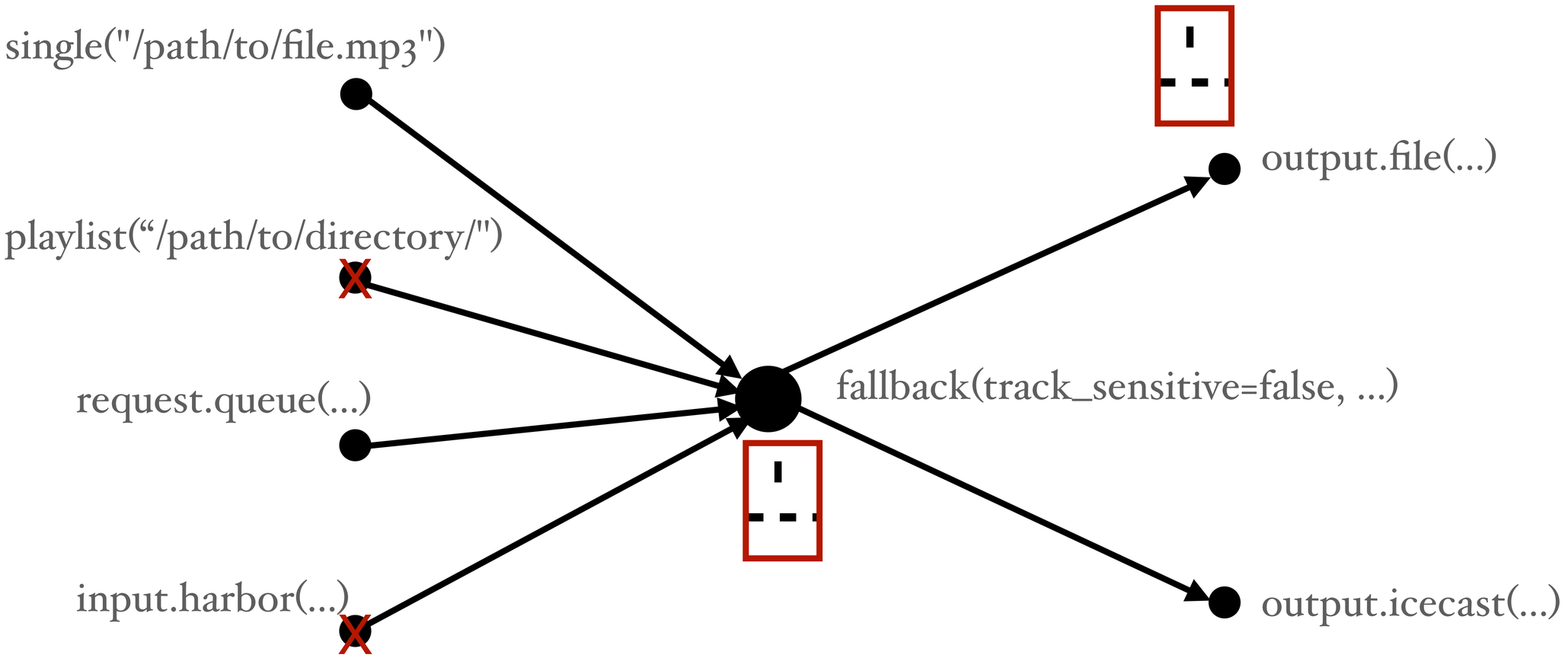
FFmpeg integration

Anything is possible!



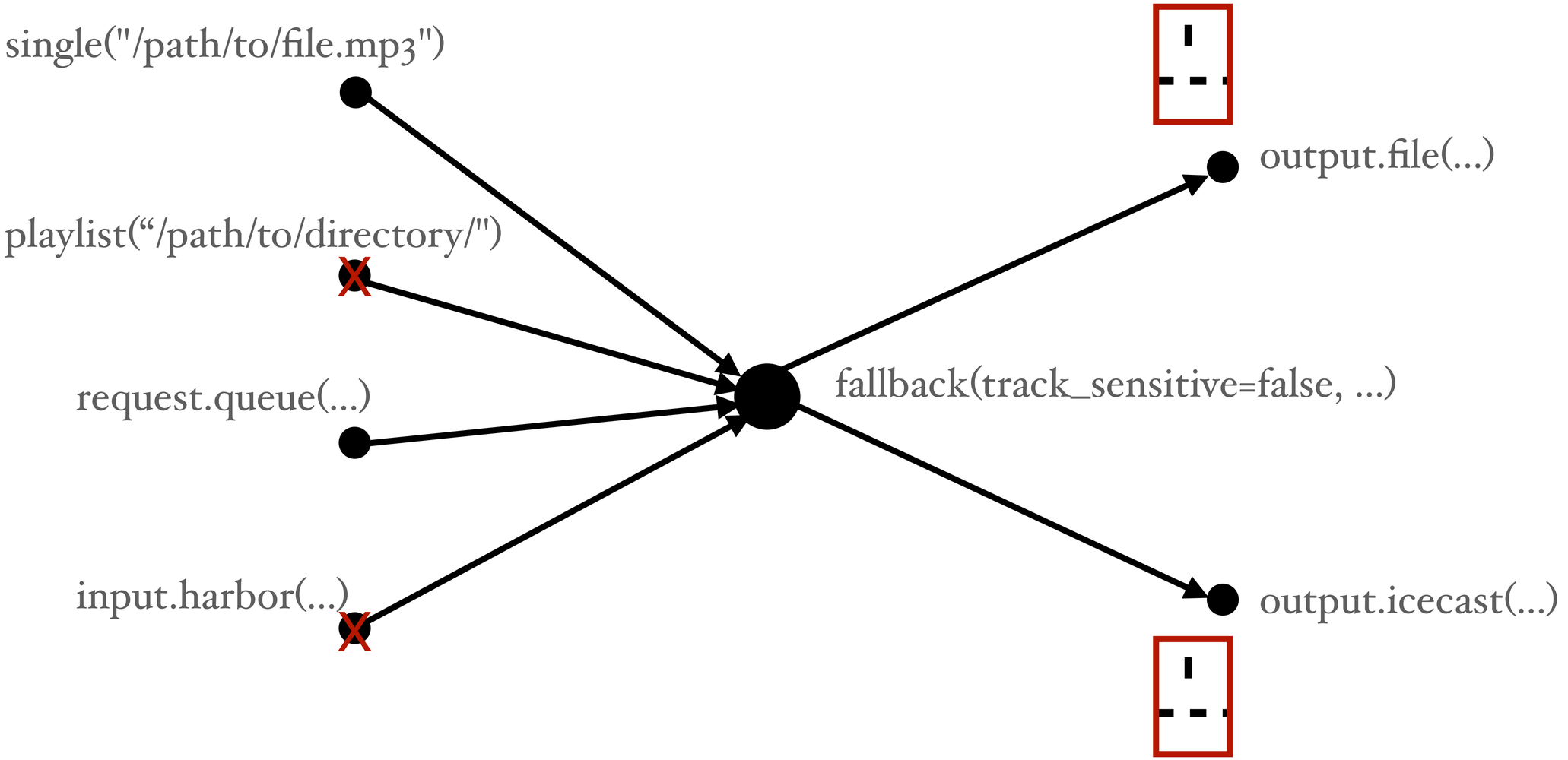
FFmpeg integration

Anything is possible!



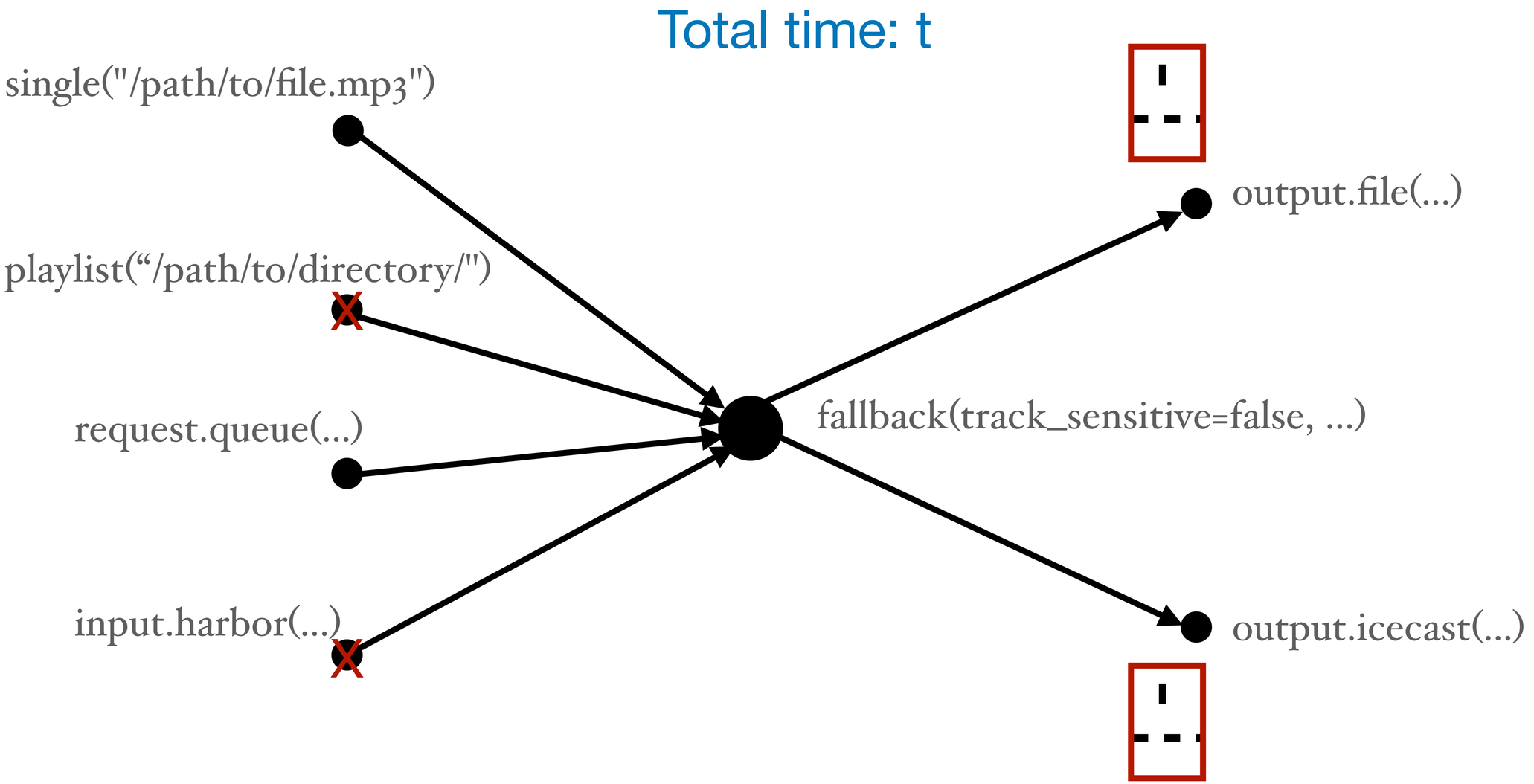
FFmpeg integration

Anything is possible!



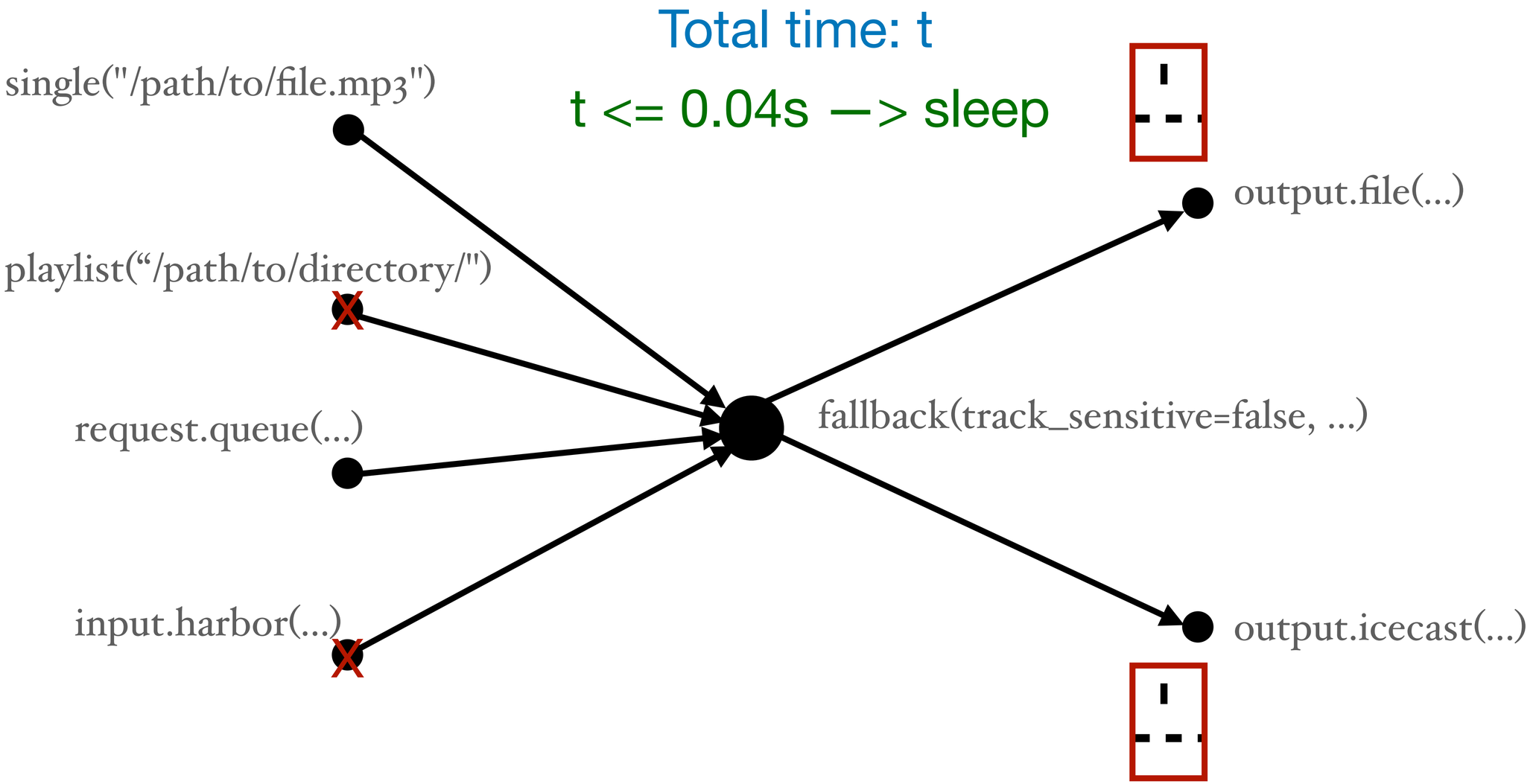
FFmpeg integration

Anything is possible!



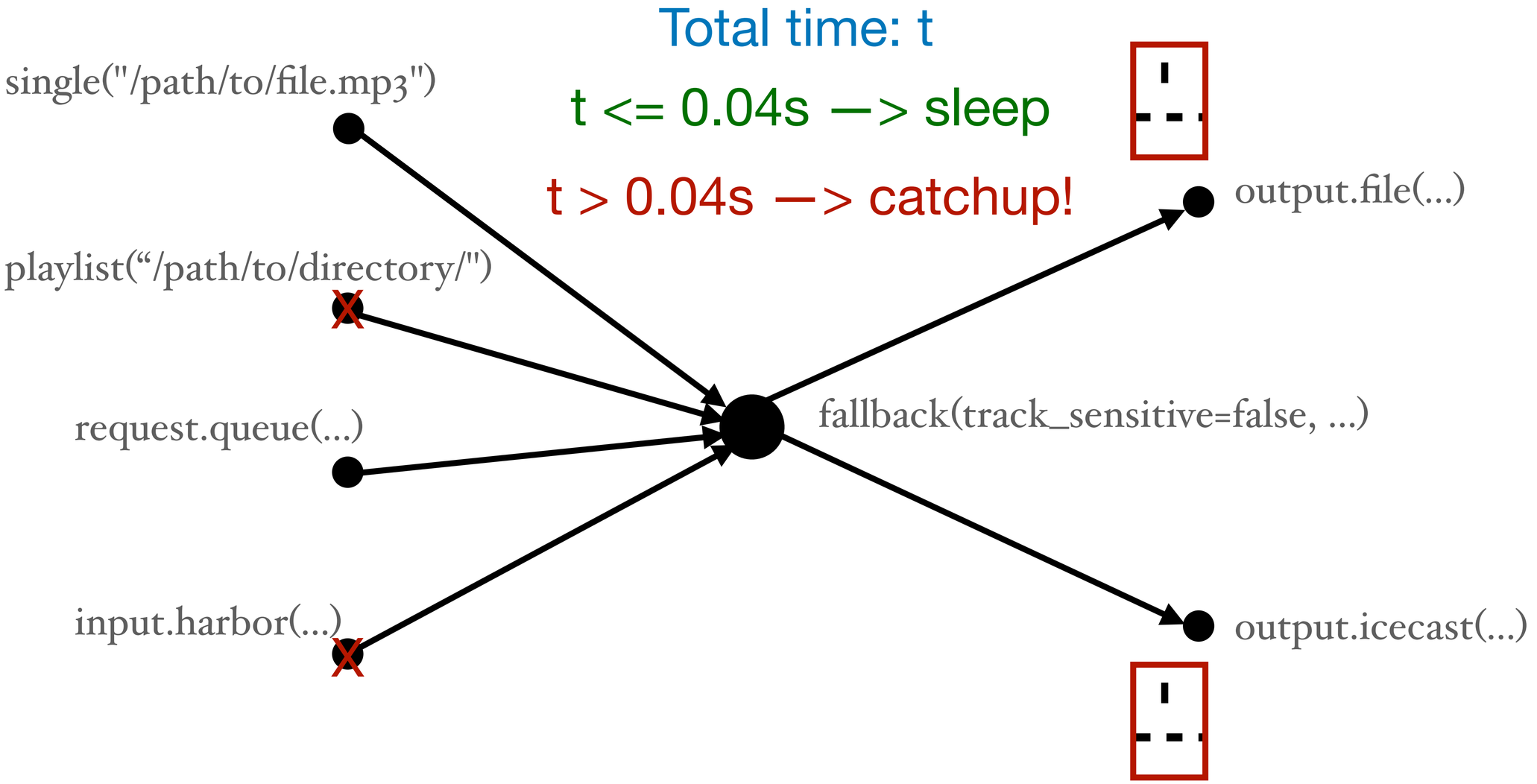
FFmpeg integration

Anything is possible!



FFmpeg integration

Anything is possible!



FFmpeg integration

Anything is possible!

FFmpeg integration

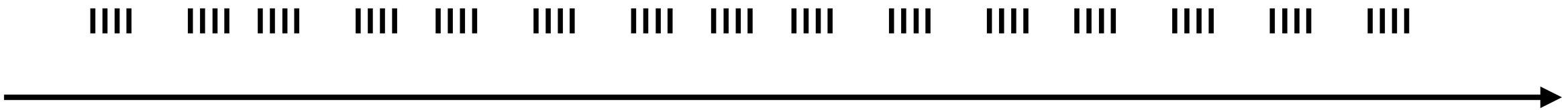
Anything is possible!

FFmpeg integration

Anything is possible!

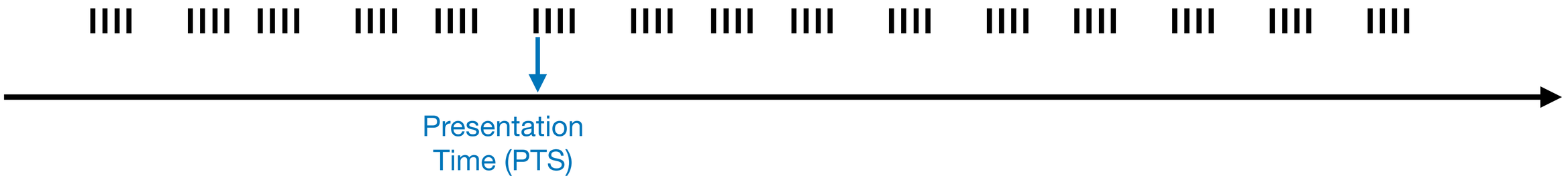
FFmpeg integration

Anything is possible!



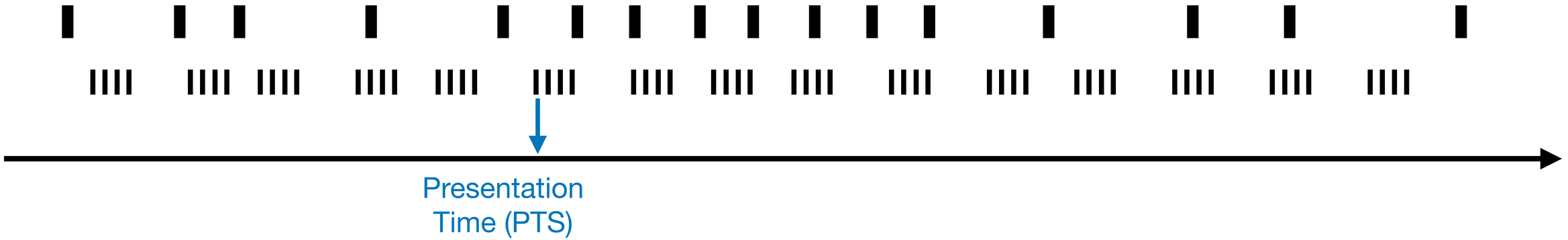
FFmpeg integration

Anything is possible!



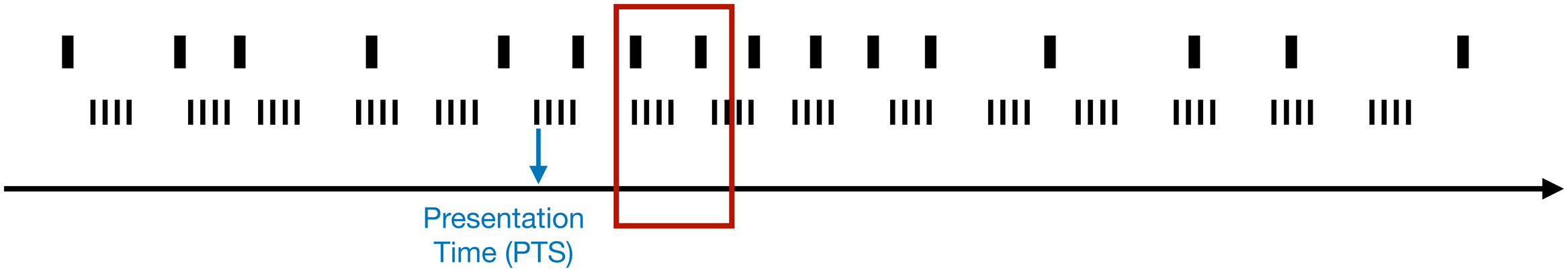
FFmpeg integration

Anything is possible!



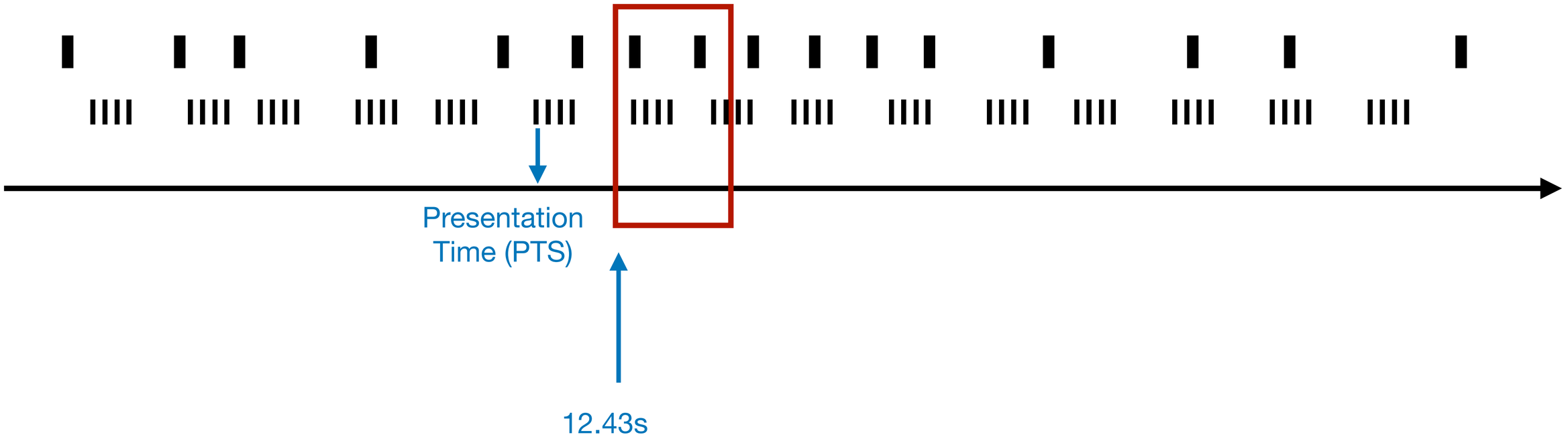
FFmpeg integration

Anything is possible!



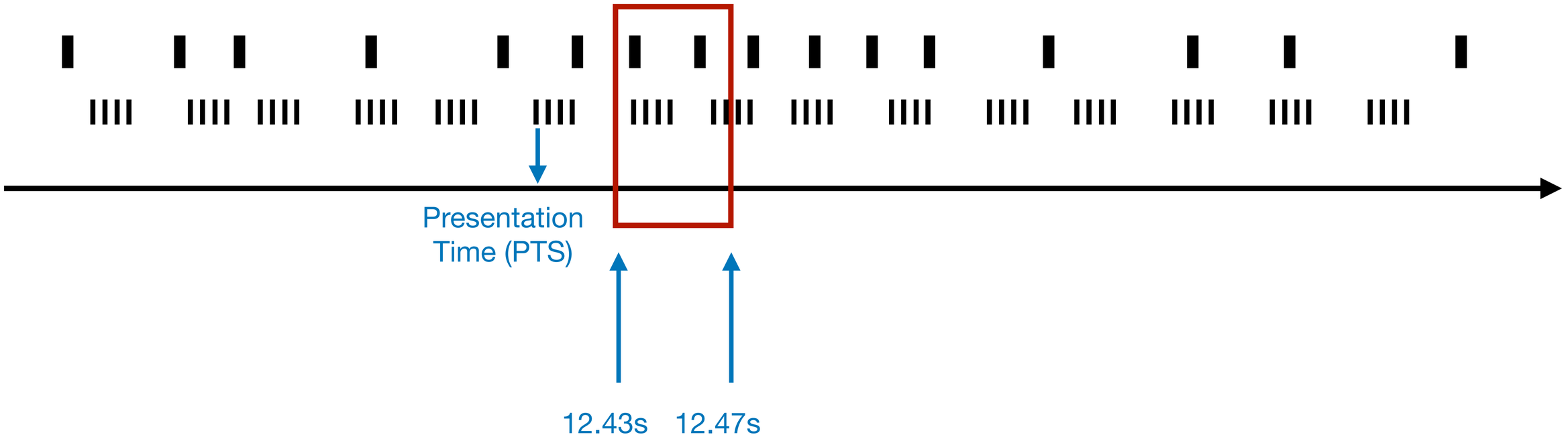
FFmpeg integration

Anything is possible!



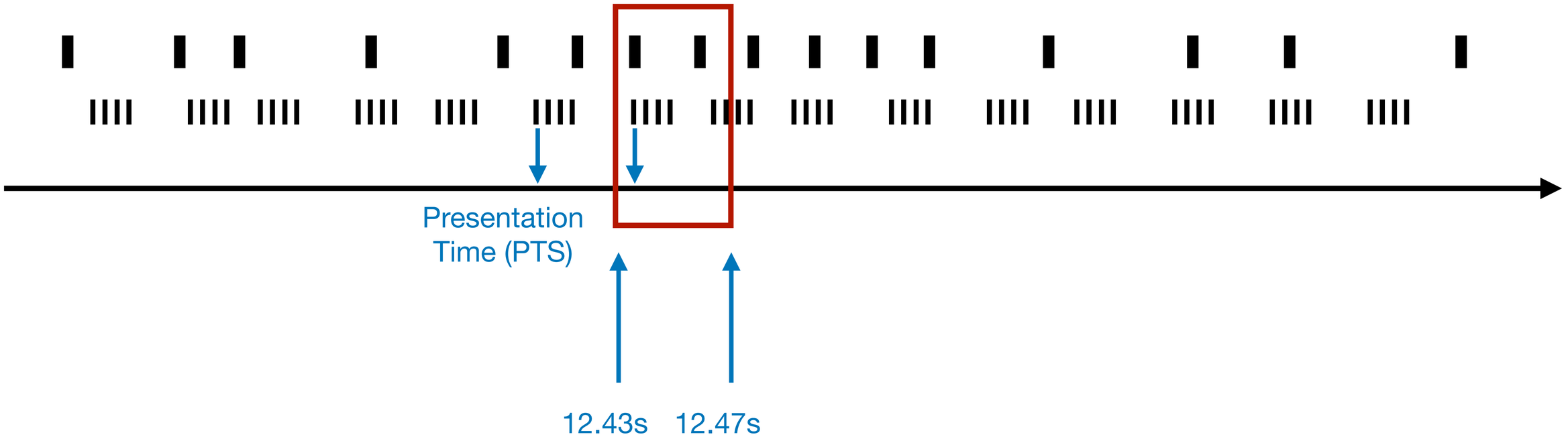
FFmpeg integration

Anything is possible!



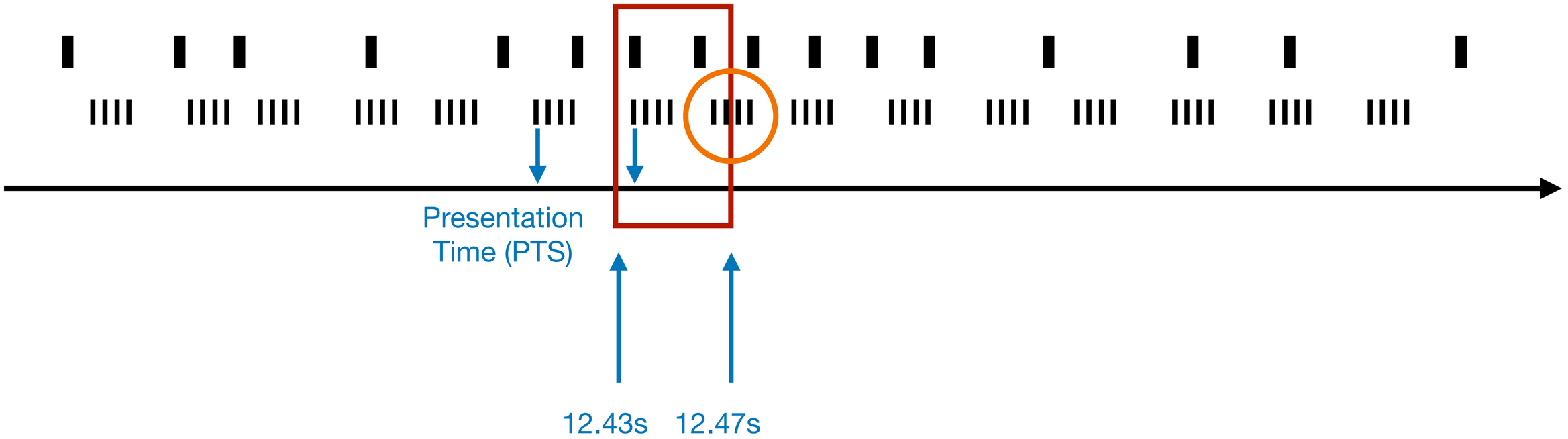
FFmpeg integration

Anything is possible!



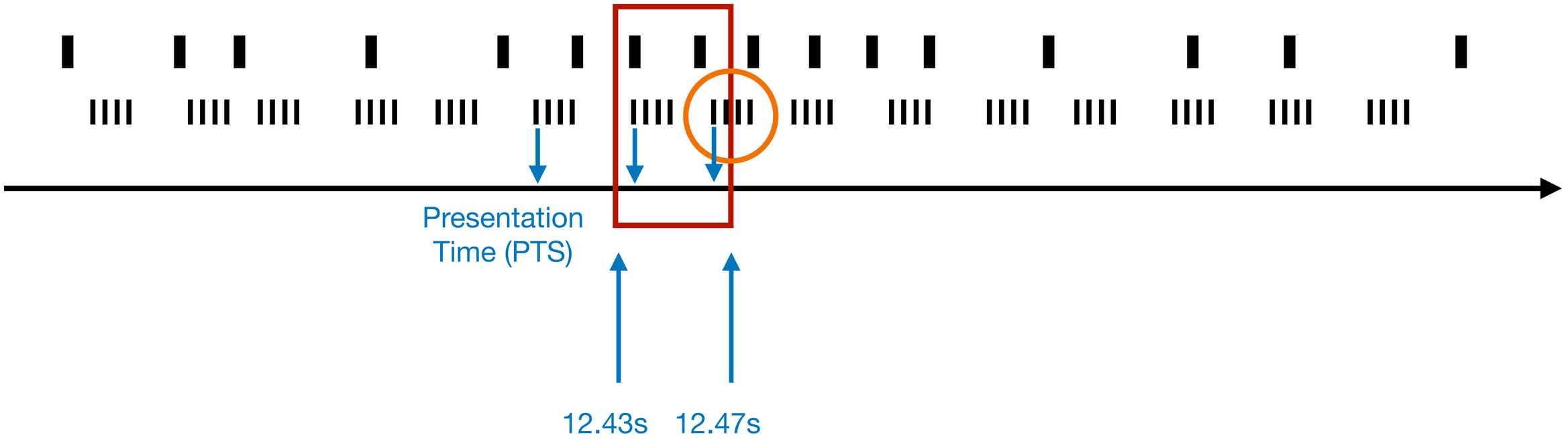
FFmpeg integration

Anything is possible!



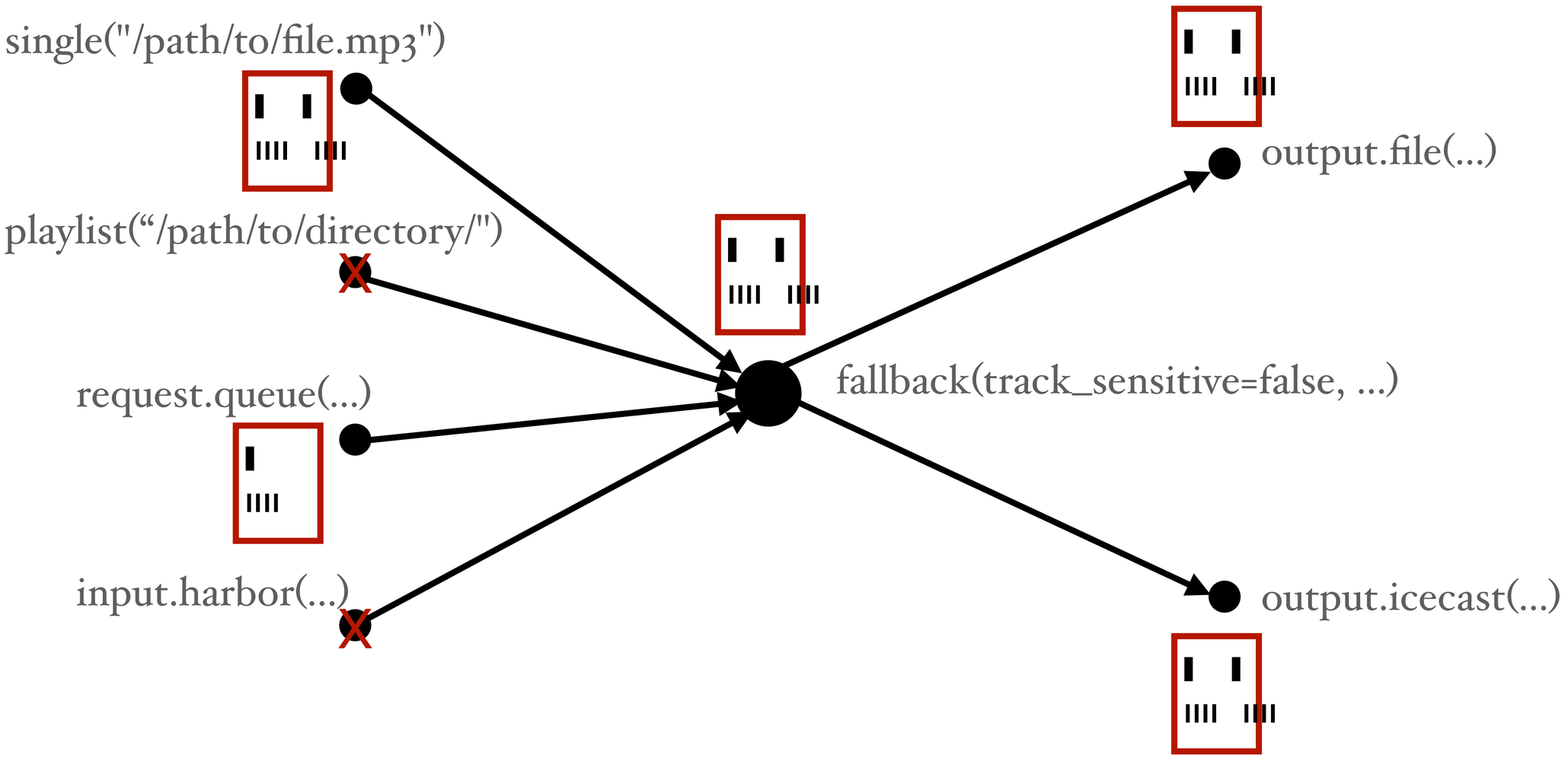
FFmpeg integration

Anything is possible!



FFmpeg integration

Anything is possible!



FFmpeg integration

Anything is possible!

FFmpeg integration

Anything is possible!

- Packet: Chunk of **encoded** data

FFmpeg integration

Anything is possible!

- Packet: Chunk of **encoded** data
  - Liquidsoap keyword: ffmpeg.copy

FFmpeg integration

Anything is possible!

- Packet: Chunk of **encoded** data
  - Liquidsoap keyword: ffmpeg.copy
- Frame: Chunk of **decoded** data

FFmpeg integration

Anything is possible!

- Packet: Chunk of **encoded** data
  - Liquidsoap keyword: ffmpeg.copy
- Frame: Chunk of **decoded** data
  - Liquidsoap keyword: ffmpeg.raw

FFmpeg integration

Anything is possible!

- Packet: Chunk of **encoded** data
  - Liquidsoap keyword: ffmpeg.copy
- Frame: Chunk of **decoded** data
  - Liquidsoap keyword: ffmpeg.raw
- Decoder

FFmpeg integration

Anything is possible!

- Packet: Chunk of **encoded** data
  - Liquidsoap keyword: ffmpeg.copy
- Frame: Chunk of **decoded** data
  - Liquidsoap keyword: ffmpeg.raw
- Decoder
- Encoder

FFmpeg integration

Anything is possible!

- Packet: Chunk of **encoded** data
  - Liquidsoap keyword: ffmpeg.copy
- Frame: Chunk of **decoded** data
  - Liquidsoap keyword: ffmpeg.raw
- Decoder
- Encoder
- Inline conversion

FFmpeg integration

Anything is possible!

- Packet: Chunk of **encoded** data
  - Liquidsoap keyword: ffmpeg.copy
- Frame: Chunk of **decoded** data
  - Liquidsoap keyword: ffmpeg.raw
- Decoder
- Encoder
- Inline conversion
- Filters

FFmpeg integration

Anything is possible!

```
%ffmpeg(
  format="mkv",
  %audio(
     codec="aac",
     channels=2,
     ar=44100
  ),
  %video(
     codec="libx264",
     b="5m",
  )
)
```

FFmpeg integration

Anything is possible!

```
%ffmpeg(
  format="mkv",
  %audio.raw(
     codec="aac",
     channels=2,
     ar=44100
  ),
  %video.raw(
     codec="libx264",
     b="5m",
  )
)
```

FFmpeg integration

Anything is possible!

```
%ffmpeg(
  format="mkv",
  %audio.copy,
  %video.copy
)
```

FFmpeg integration

Anything is possible!

```
%ffmpeg(
  format="mkv",
  %audio.copy,
  %video.copy
)
```

```
s = input.harbor("mount")

output.icecast(
  %ffmpeg(format=“mp3", %audio.copy),
  server="host1",
  …,
  s
)
```

FFmpeg integration

Anything is possible!

```
%ffmpeg(
  format="mkv",
  %audio.copy,
  %video.copy
)
```

```
s = input.harbor("mount")

output.icecast(
  %ffmpeg(format="mp3", %audio.copy),
  server="host1",
  …,
  s
)

output.icecast(
  %ffmpeg(format="mkv", %audio.copy, %video.copy),
  server="host2",
  …,
  s
)
```

FFmpeg integration

Anything is possible!

```
%ffmpeg(
  format="mkv",
  %audio.copy,
  %video.copy
)
```

```
s = input.harbor("mount")

output.icecast(
  %ffmpeg(format="mp3", %audio.copy),
  server="host1",
  …,
  s
)

output.file.hls(
  streams=[
    ("mp3", %ffmpeg(format="mp3", %audio.copy))
  ],
  …,
  s
)
```

FFmpeg integration

Anything is possible!

```
source(
  audio=ffmpeg.audio.copy('a),
  video=none,
  midi=none)
```

FFmpeg integration

Anything is possible!

```
source(
  audio=ffmpeg.audio.copy('a),
  video=none,
  midi=none)

[decoder.ffmpeg:4] ffmpeg recognizes "/tmp/bla.mp4" as:
    audio: {codec: aac, 48000Hz, 2 channel(s)},
    video: {codec: h264, 1920x1080, yuv420p}
```

FFmpeg integration

Anything is possible!

```
source(
  audio=ffmpeg.audio.copy('a),
  video=none,
  midi=none)

[decoder.ffmpeg:4] ffmpeg recognizes "/tmp/bla.mp4" as:
    audio: {codec: aac, 48000Hz, 2 channel(s)},
    video: {codec: h264, 1920x1080, yuv420p}

[/tmp/bla(dot)mp4:4] Content type is {
  audio=ffmpeg.audio.copy(
    sample_rate=48000,sample_format=fltp,channel_layout=“stereo”,codec="aac"
   ),
  video=none,midi=none
}
```

FFmpeg integration

Anything is possible!

liquidsoap -h ffmpeg.encode.audio_video

Convert a source's content

```
Type: (
  ?id : string,
  ?buffer : float,
  ?max : float,
  format(audio=pcm('a), video=yuva420p('b), midi=none),
  source(audio=pcm('a), video=yuva420p('b), midi=none)
) ->
  source(audio=ffmpeg.audio.copy(‘e), video=ffmpeg.video.copy(‘f), midi=none)
```

FFmpeg integration

Anything is possible!

```
s = (...)

s = ffmpeg.encode.audio(
  %ffmpeg(
    format="mp3",
    %audio(codec="libmp3lame", b="128k")
  )
)
```

FFmpeg integration

Anything is possible!

```
s = (...)

s = ffmpeg.encode.audio(
  %ffmpeg(
    format="mp3",
    %audio(codec="libmp3lame", b="128k")
  )
)
```

```
output.icecast(
  %ffmpeg(format="mp3", %audio.copy)
  server="host1",
  …,
  s
)

output.icecast(
  %ffmpeg(format="mp3", %audio.copy)
  server="host2",
  …,
  s
)
```

FFmpeg integration

Anything is possible!

```
s = (...)

s = ffmpeg.encode.audio_video(
  %ffmpeg(
    format="mkv",
    %audio(
      codec="aac",
      b="128k",
    ),
    %video(
      codec="libx264",
      b="5M"
    )
  )
)
```

FFmpeg integration

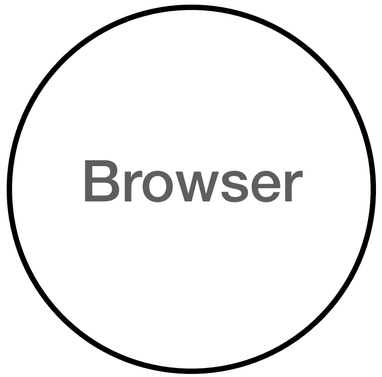
Anything is possible!

```
s = (...)

s = ffmpeg.encode.audio_video(
  %ffmpeg(
    format="mkv",
    %audio(
      codec="aac",
      b="128k",
    ),
    %video(
      codec="libx264",
      b="5M"
    )
  )
)
```

```
mpegts = %ffmpeg(format="mpegts", %audio.copy, %video.copy)

mp4 = %ffmpeg(
        format="mp4",
        movflags="+dash+skip_sidx+skip_trailer+frag_custom",
        frag_duration=10,
        %audio.copy, %video.copy
)

streams = [
  ("mpegts", mpegts),
  ("mp4", mp4)
]

output.file.hls(streams, …, s)
```

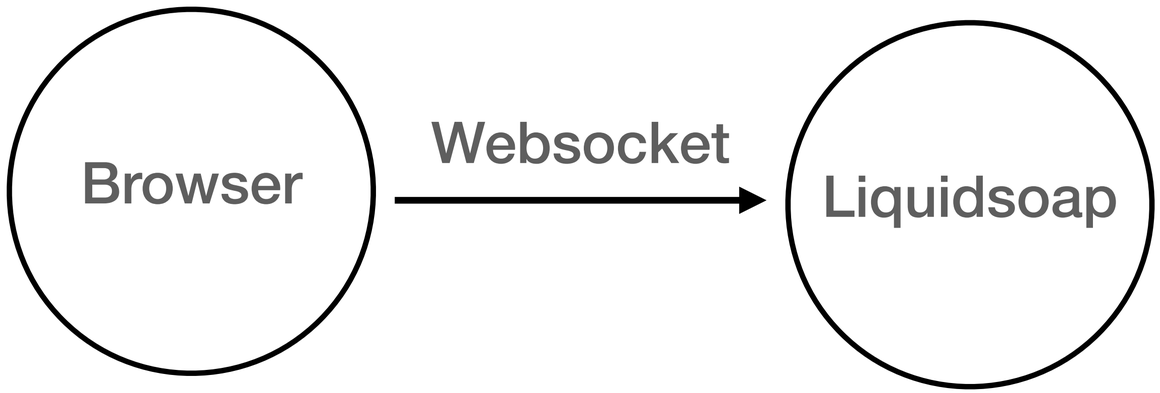
FFmpeg integration

Anything is possible!



FFmpeg integration

Anything is possible!

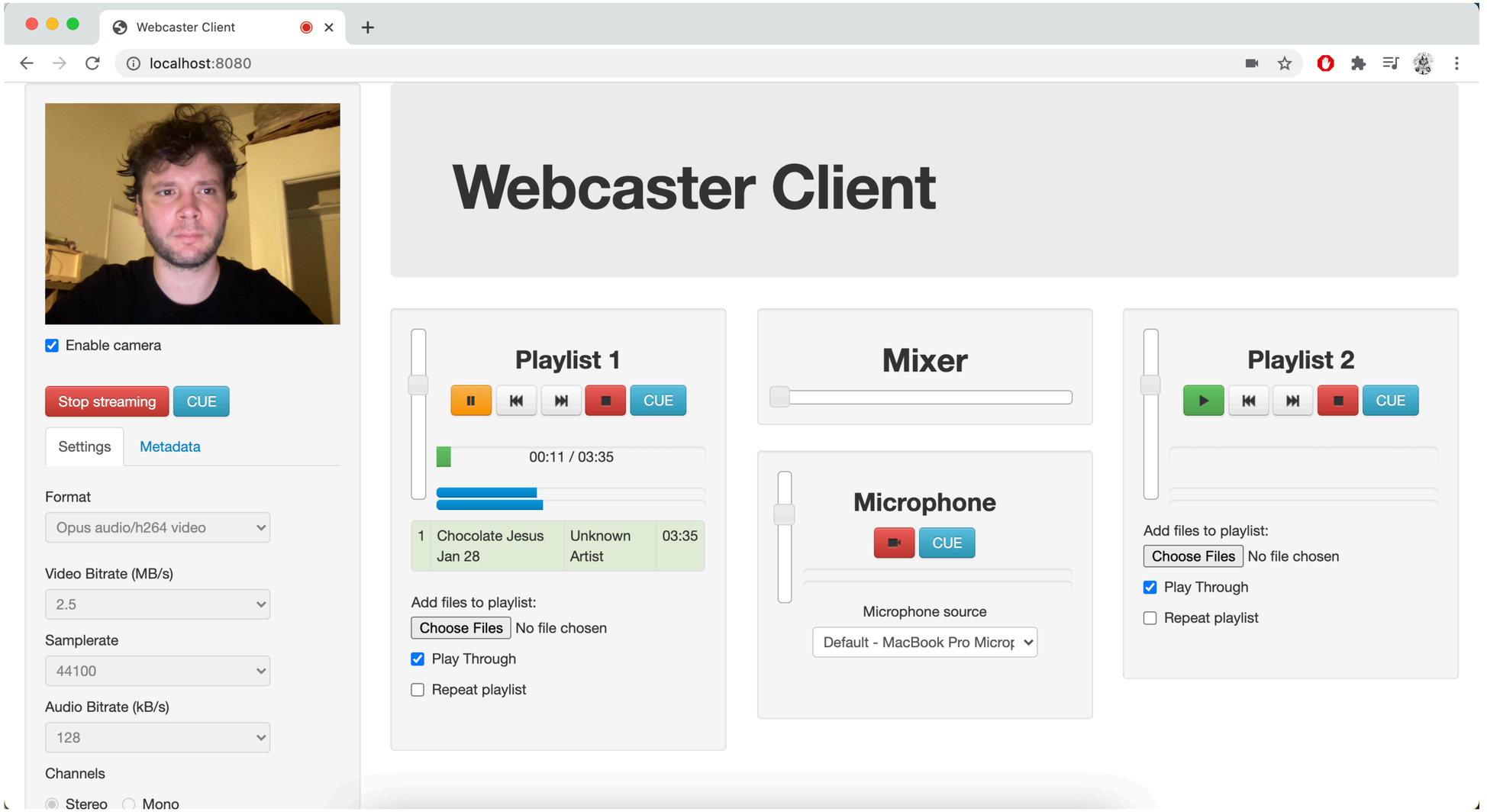


FFmpeg integration

Anything is possible!

FFmpeg integration

Anything is possible!



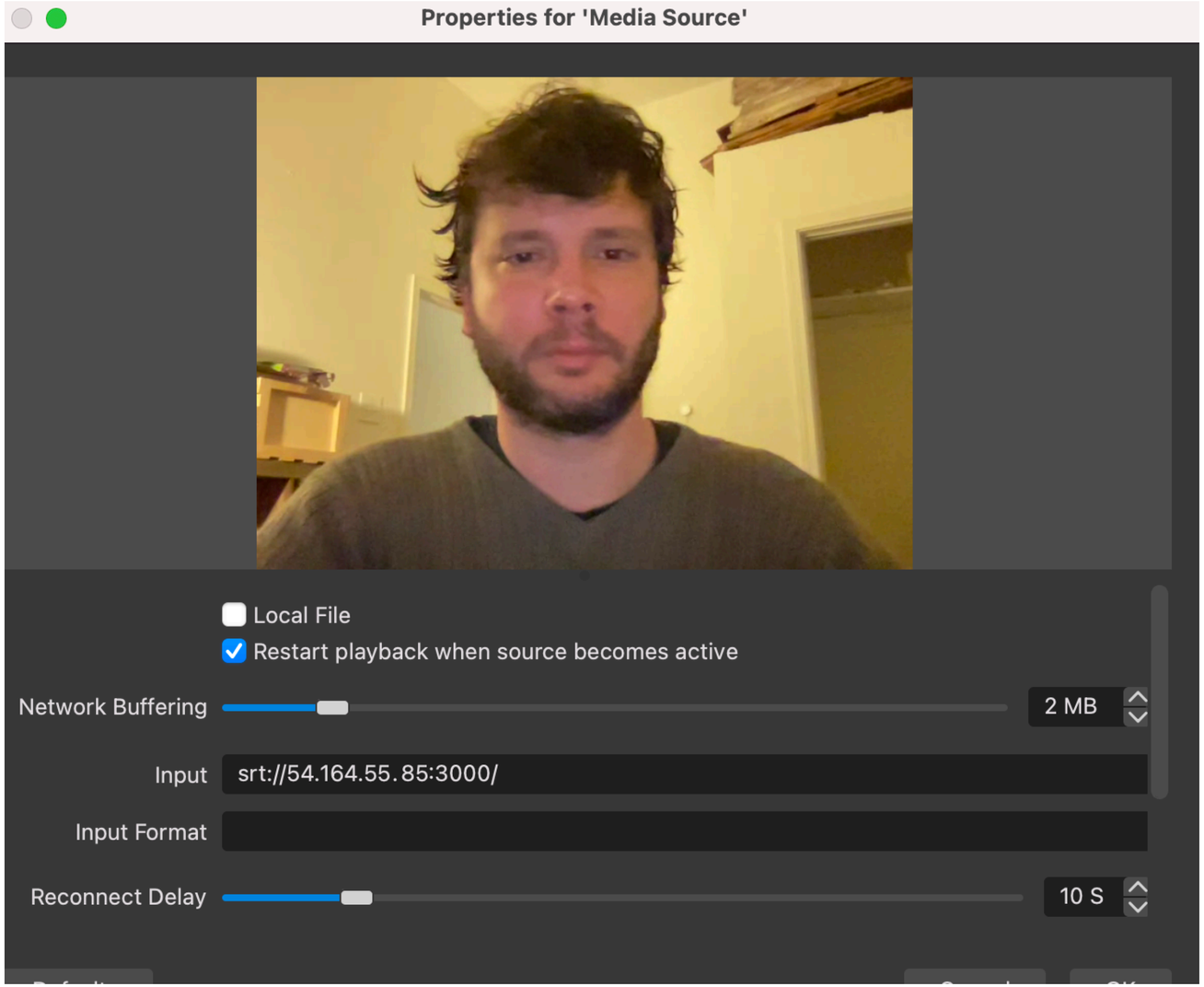
FFmpeg integration

Anything is possible!

```
s = input.harbor(port=8000, "mount")

e = %ffmpeg(format="webm",%audio.copy, %video.copy)

output.srt(mode="listener", port=3000, e, fallible=true, s)
```

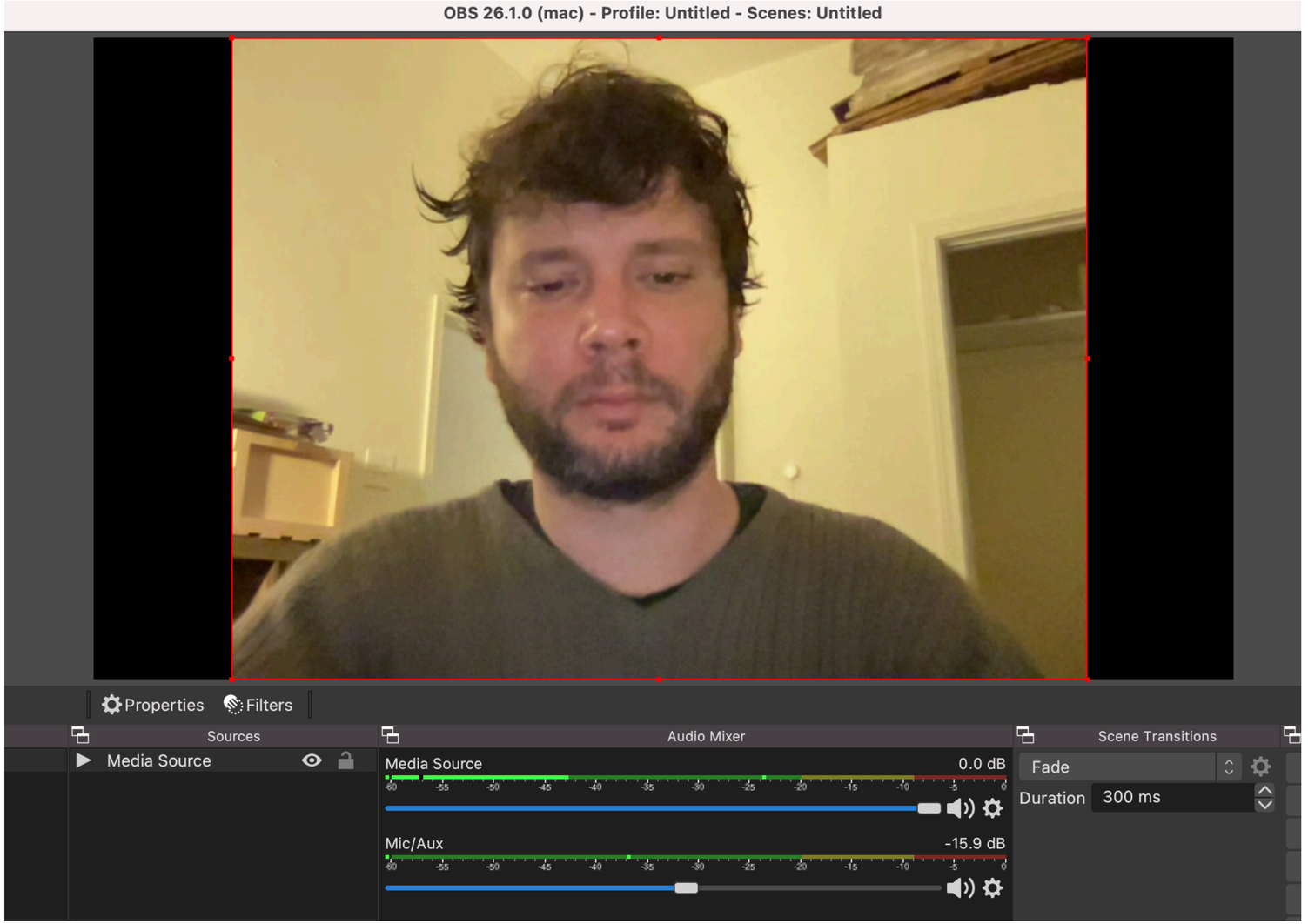
FFmpeg integration

Anything is possible!



FFmpeg integration

Anything is possible!



TODO

What’s next?

TODO

What’s next?

- Cleanup, fix bugs

TODO

What’s next?

- Cleanup, fix bugs
- Re-organize the video API with modules

TODO

What’s next?

- Cleanup, fix bugs
- Re-organize the video API with modules
- Complete overhaul of the documentation

TODO

What's next?

- Cleanup, fix bugs
- Re-organize the video API with modules
- Complete overhaul of the documentation
- RTMP input?

TODO

What's next?

- Cleanup, fix bugs
- Re-organize the video API with modules
- Complete overhaul of the documentation
- RTMP input?
- Implement awesome features:

TODO

What's next?

- Cleanup, fix bugs
- Re-organize the video API with modules
- Complete overhaul of the documentation
- RTMP input?
- Implement awesome features:
  - SRT HEVC server

TODO

What's next?

- Cleanup, fix bugs
- Re-organize the video API with modules
- Complete overhaul of the documentation
- RTMP input?
- Implement awesome features:
  - SRT HEVC server
  - …