# adm-toolbox

EBU

Feb 03, 2023

# CONTENTS:

The EAT is a set of tools for processing ADM (Audio Definition Model) files. It can convert ADM files between profiles, validate them, render them, fix common issues, and more.

It contains:

- A *framework* for building processing graphs that operate on audio and associated data.
- A *set of ADM-related processes* that sit within this framework.
- A *command-line tool* `eat-process`, which processes ADM files using the processes from the framework, as defined in a *configuration file*.
- A set of *example configuration files* for the tool, for doing things like profile conversion, loudness measurement, fixing common issues, validation etc.

See the README file on github for installation instructions and license information.

CHAPTER

ONE

# FRAMEWORK

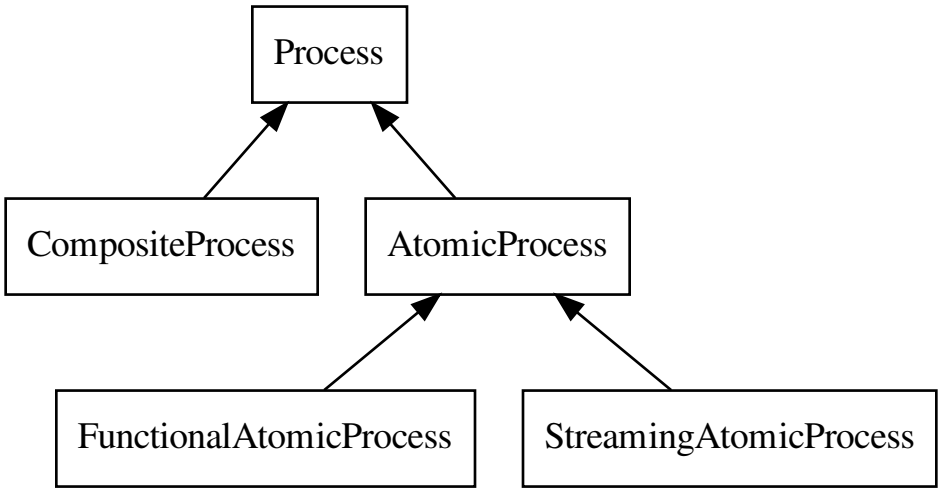
The framework of the ADM Toolbox provides a structure for components which process ADM files to fit into.

This takes the form of a *processing graph*: The individual components are *processes*, which have input and output *ports* through which they communicate. These can be connected together in a *graph* structure, which is a collection of processes, and connections between their ports.

For example, consider a "read BW64" process with output ports for audio samples and an ADM document, and a "write BW64" process with input ports for audio samples and an ADM document. These may be connected together to form a kind of BW64 copy operation.

## 1.1 Processes

There are several kinds of process, shown in the following inheritance diagram:




`Process` is the base class for all process types. It has input and output ports, and can be added to graphs.

This is split into two types: `AtomicProcess` and `CompositeProcess`. Atomic processes actually implement some processing (i.e. they are not divisible), while composite processes just contain other processes and connections between them, which may be themselves be composite or atomic.

Atomic processes are further divided into two types: *FunctionalAtomicProcess* and *StreamingAtomicProcess*.

In a functional processes, the outputs are a function of the inputs: they implement a *process()* method, which is called once, and should read from the input ports and write to the output ports.

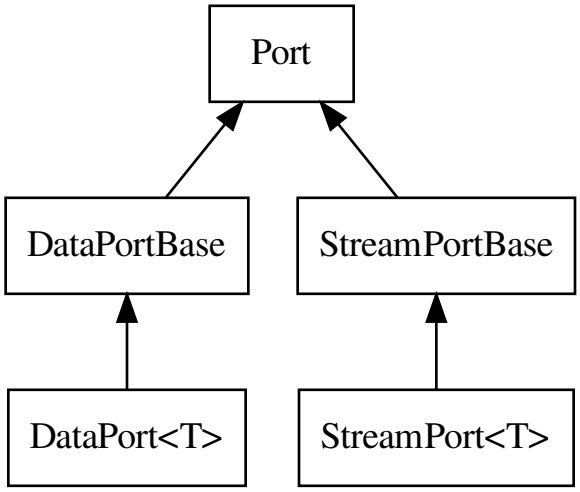
While functional processes are limited to operating on single pieces of data, streaming processes operate on streams of data. They implement three methods:

- *initialise()* is called once, and can read from non-streaming input ports
- *process()* should read from streaming input ports and write to streaming output ports, and is called repeatedly as long as any ports are not closed (see below)
- *finalise()* is called once, and can write to non-streaming output ports

For example, a loudness meter would be a streaming process: `process` would read audio samples from a streaming input port, performing the analysis. and the accumulated loudness values would be written to a non-streaming output port in `finalise`.

## 1.2 Ports

As aluded to above, processes can have two kinds of ports, *data ports* and *streaming ports*. Additionally, each port has a type, and can only be connected to ports of the same type. This is shown in this inheritance diagram:




*Port* can be used to reference any type of port, and is mainly used for making connections. *DataPort* and *StreamPort* are concrete ports of a particular type, and are mostly used inside processes. *DataPortBase* and *StreamPortBase* are interfaces used by the implementation.

Data ports hold a value of the given type. A process writing to a data port should use *set_value()*, while a process reading from a data port should use *get_value()*.

The framework moves or copies this value between connected ports.

Stream ports hold a queue of values of the given type, and an `eof` (End Of File) flag.

Writers should call *push()* to write items, followed by *close()* to signal that no more items will be pushed.

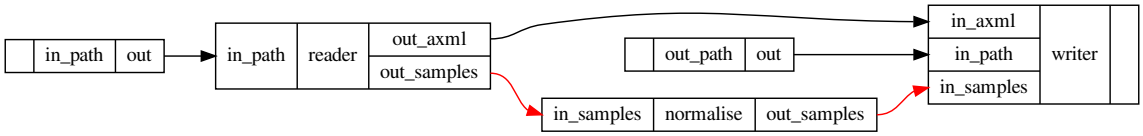
Readers should use *available()* to see if there are any items in the queue, and *pop()* to read an item. When *eof()* returns true, there are no items left to read, and the writer has closed the port.

The framework moves or copies items and the `eof` flag between ports.

See *Port Value Semantics* for more detail on how data is transferred between ports.

## 1.3 An Example

The graph below is for an application which normalises the levels in an audio file while retaining ADM metadata:

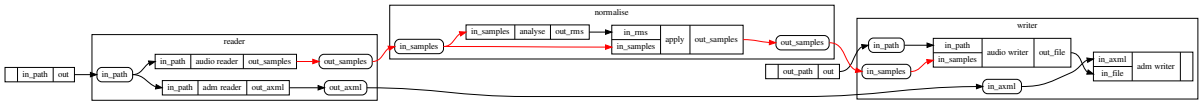



Red lines represent streaming connections. Processes are shown as columns with the input ports on the left, output ports on the right, and the process name in the middle.

The components are as follows:

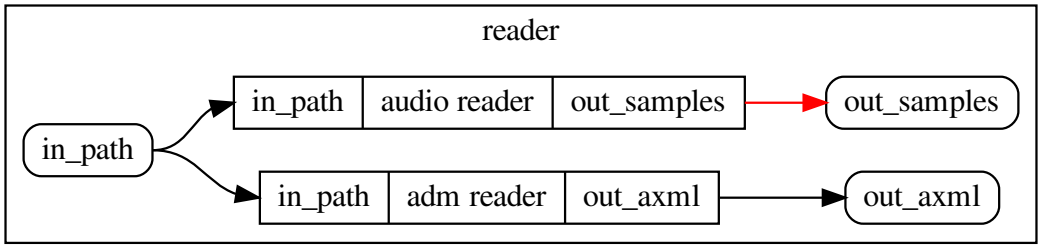
- *in_path* and *out_path* are `DataSource` processes which produce the input and output path name
- *reader* is a composite process which reads ADM metadata and samples (streaming) from a BW64 file
- *writer* is a composite process which writes ADM metadata and samples (streaming) to a BW64 file.
- *normalise* is a composite process which normalises its input samples to produce some output samples.

If these composite processes are expanded it looks like this (you may have to open in a new tab…):



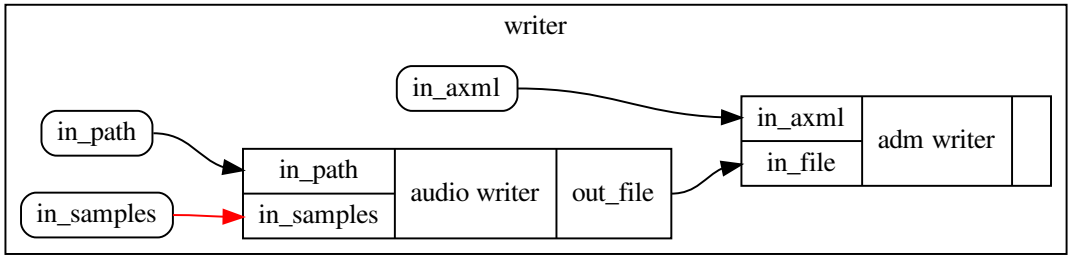

Here, composite processes are shown as boxes containing their constituent processes, with rounded boxes representing their input and output ports (due to the limitations of graphviz).

Zooming in, *reader* looks like this:



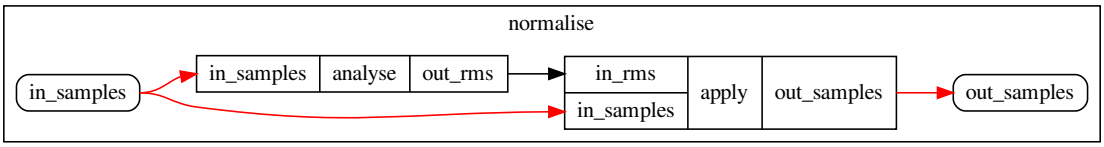

It consists of two independent processes which read the samples and ADM data, so there is no ordering constraint between them.

Writer is more complex:




libbw64 does not support editing files, so the samples and ADM metadata need to be written using the same `Bw64Writer` object. To do this, the *audio writer* process sends the writer object out of a port, which is used by the *adm writer* process. These could technically be merged into one atomic process, but this way the ADM metadata does not have to be available before the samples.

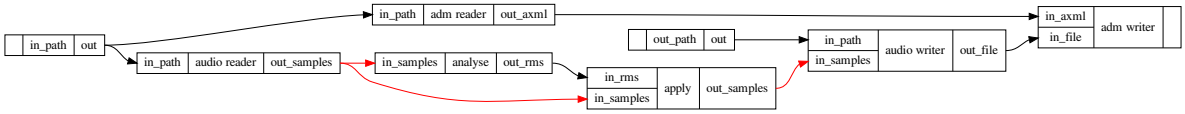
The *normalise* process looks like this:




The *analyse* process takes streaming audio and measures the RMS level of the whole of each channel; these are produced on a data port. These RMS levels are used by the *apply* process to modify the level of the input samples.

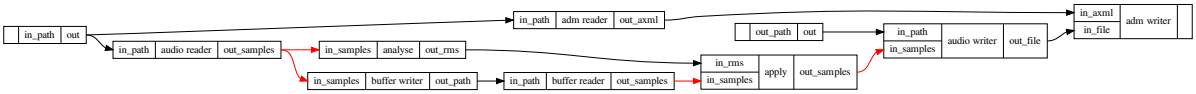
### 1.3.1 Evaluation

To evaluate the graph, the first step is to flatten it, expanding composite processes:




This exposes a problem: there is a streaming connection from *audio reader* to *analyse* and *apply*, but there's a non-streaming connection between *analyse* and *apply*. Because non-streaming ports are read before streaming and written after streaming (see `StreamingAtomicProcess`), it's not possible to stream between all three processes simultaneously.

To deal with this situation, *buffer writer* and *buffer reader* processes are automatically inserted to split enough streaming connections that this does not occur. The graph then looks like this:




Now *audio reader*, *analyse* and *buffer writer* can run together, followed by *buffer reader*, *apply* and *audio writer*, because there are no non-streaming connections within each of these sub-graphs.

The type of buffer writer and reader used can be specialised for each type of streaming port by specialising `MakeBuffer`. The default implementation buffers stream values into a `std::vector`, which defeats the memory savings of streaming. A specialisation is provided for audio samples which writes to a temporary wav file instead.

## 1.4 Port Value Semantics

One output port may be connected to multiple input ports. To implement this, the value stored in the port is copied to all but the last connected port, and moved to the last output port.

Thus, the data stored in a port should have value semantics – that is a copy creates a new value with the same contents, and changing one copy does not affect other copies. This is done because it's much easier to implement sensible reference semantics on top of value semantics, than it is the other way around.

Basic types (ints, floats etc.), POD types and STL containers meet this criteria, while `std::shared_ptr` and some custom classes do not.

To work with types like libadm documents which are always accessed through a `shared_ptr`, `ValuePtr` is provided. This allows each reader to chose whether they want a const or a non-const pointer, which can save copying the document in cases where all readers access the value through a const pointer, or there is only one reader which access the value through a non-const pointer.

## 1.5 Other Features

This section is lists features that the framework is designed to support, but are not currently implemented.

### 1.5.1 Progress

When processing large files it would be nice to indicate the progress to the user. There are two parts to this:

- Each *ExecStep* in a *Plan* represents one step of the evaluation. These should provide more information about what they are doing (e.g. a name and a list of processes it will run) so that the overall process through the graph can be reported.
- Streams should be able to optionally report their progress as a percentage. Often there will be just one process in a streaming sub-graph that knows how far through it is (e.g. a file reader), and this can be reported through a callback.

### 1.5.2 Streaming ADM

A streaming ADM BW64 file can be thought of as a sequence of ADM documents with associated ranges of samples. To process these within this framework, one solution would be to allow the graph to run multiple times (once on each document). This should allow components to be shared between streaming and non-streaming uses.

### 1.5.3 Duplicating Streaming Processes

The example in the last section is wasteful, in that the samples from the original file are written to and read from a temporary file in order to break a streaming connection – It would be better if the original file could be read a second time.

Processes should be able to be specify that they are safe to copy (i.e. will always produce the same output given the same inputs with no side-effects), and the framework should prefer to use this if possible to break streaming connections.

### 1.5.4 Exceptions

Processes can raise exceptions while running. Currently these are just propagated to the user, but should be wrapped in another exception that records which process they came from in order to improve error reporting.

## 1.6 Writing Processes

This section briefly explains how to write some different kinds of processes.

### 1.6.1 Functional Atomic Process

An example of a process that adds 1 to an integer input:

```
class AddOne : public FunctionalAtomicProcess {
 public:
  AddOne(const std::string &name)
      : FunctionalAtomicProcess(name),
        in(add_in_port<DataPort<int>>("in")),
        out(add_out_port<DataPort<int>>("out")) {}

  virtual void process() override {
    out->set_value(in->get_value() + 1);
  }

 private:
  DataPortPtr<int> in;
  DataPortPtr<int> out;
};
```

Note that:

- The process name is passed in through the constructor, and should normally be passed straight to the *FunctionalAtomicProcess* constructor.
- Ports are added through *Process::add_in_port()* and *Process::add_out_port()*, with the port type as a parameter. These are saved (as the corresponding pointer type) for use in process.
- *DataPort::get_value()* and *DataPort::set_value()* are used to get and set the values of input and output port respectively.

For heavier types, `std::move()` should be used in *process()* like this example with a std::vector input and output:

```
virtual void process() override {
  // move from the input port to avoid copying the data
  std::vector<int> value = std::move(in->get_value());

  // modify it
  value.push_back(7);

  // move to the output port to avoid copying again
  out->set_value(std::move(value));
}
```

For types using *ValuePtr*, it will look something like this for modifying a value in-place:

```
virtual void process() override {
  // get the wrapper
  ValuePtr<std::vector<int>> value_ptr = std::move(in->get_value());

  // extract the value; this will copy or move if it's the last user
  std::shared_ptr<std::vector<int>> value = value_ptr.move_or_copy();

  // modify it
  value->push_back(7);
```

(continues on next page)

(continued from previous page)

```
  // move to the output port
  out->set_value(std::move(value));
}
```

Or this if read-only access is OK:

```
virtual void process() override {
  // get the wrapper
  ValuePtr<std::vector<int>> value_ptr = std::move(in->get_value());

  // extract a reference to the value
  std::shared_ptr<const std::vector<int>> value = value_ptr.read();

  // use it somehow
  out->set_value(value->at(0));
}
```

### 1.6.2 Streaming Atomic Process

An example of a process that produces a stream that's the same as the input, but one greater:

```
class AddOneStream : public StreamingAtomicProcess {
 public:
  AddOneStream(const std::string &name)
      : StreamingAtomicProcess(name),
        in(add_in_port<StreamPort<int>>("in")),
        out(add_out_port<StreamPort<int>>("out")) {}

  virtual void process() override {
    while (in->available())
      out->push(in->pop() + 1);

    if (in->eof())
      out->close();
  }

 private:
  StreamPortPtr<int> in;
  StreamPortPtr<int> out;
};
```

Note that:

- All ports must be empty and closed (check with *StreamPort::eof()*) for the streaming to finish. Generally output ports should be closed once corresponding inputs have ended (*StreamPort::eof()*). It's valid to close a port multiple times (it has no effect), so there's no need to track if it's been closed or not.
- *StreamPort::push()* and *StreamPort::pop()* behave similarly to *DataPort::set_value()* and *DataPort::get_value()*, so the above information about `std::move()` and *ValuePtr* apply here too.
- *StreamPort::pop()* will throw an exception if there's nothing in the queue, so use *StreamPort::available()*.

- To get data from input data ports or send data to output data ports, override *initialise()* or *finalise()* respectively.

### 1.6.3 Composite Process

Here's a composite process which chains together two `AddOne` processes defined earlier:

```
class AddTwo : public CompositeProcess {
 public:
  AddTwo(const std::string &name)
      : CompositeProcess(name) {
    // add ports for this process
    auto in = add_in_port<DataPort<int>>("in");
    auto out = add_out_port<DataPort<int>>("out");

    // add sub-processes
    auto p1 = add_process<AddOne>("p1");
    auto p2 = add_process<AddOne>("p2");

    // connect everything together
    connect(in, p1->get_in_port("in"));
    connect(p1->get_out_port("out"), p2->get_in_port("in"));
    connect(p2->get_out_port("out"), out);
  }
};
```

- Ports are added using the same functions as for atomic processes.
- Sub-processes are added using *Graph::add_process()* or *Graph::register_process()*.
- Ports are connected using *Graph::connect()*. *Process::get_out_port()* and *Process::get_in_port()* are used to access ports of sub-processes by name. All external ports, and ports of sub-processes must be connected.
- There's no need to keep a reference to the ports or processes.

## 1.7 Building Processing Graphs

Processing graphs can be built with the *Graph* class, and ran using *evaluate()*, like this:

```
Graph g;

auto p1 = g.add_process<AddOne>("p1");
auto p2 = g.add_process<AddOne>("p2");

// connect processes together
g.connect(p1->get_out_port("out"), p2->get_in_port("in"));

// add input
auto in = g.add_process<DataSource<int>>("input", 5);
g.connect(in, p1->get_in_port("in"));
```

(continues on next page)

(continued from previous page)

```
// add output
auto out = g.add_process<DataSink<int>>("output");
g.connect(p2->get_out_port("out"), out);

// run the graph
evaluate(g);

// check the output
assert(out->get_value() == 7);
```

This is exactly the same API as is used for building composite processes.

Again, all ports of all processes mus be connected. Inputs and outputs can accessed by adding and connecting *DataSource* and *DataSink* processes, and unused output ports can be terminated with *NullSink* processes.

CHAPTER

TWO

# CONFIGURATION FILE

EAT processing pipelines (see `framework`) can be configured through a JSON file, which specifies a list of processes to run, and connections between them.

A basic example might look something like this:

```
{
  "version": 0,
  "processes": [
    {
      "name": "input",
      "type": "read_adm_bw64",
      "parameters": {
        "path": "/tmp/in.wav"
      },
      "out_ports": ["out_axml", "out_samples"]
    },
    {
      "name": "output",
      "type": "write_adm_bw64",
      "parameters": {
        "path": "/tmp/out.wav"
      },
      "in_ports": ["in_axml", "in_samples"]
    }
  ]
}
```

At the root there are three possible keys:

- `version`

  The version number of the configuration file format, to ensure compatibility with future versions of the EAT.

- `processes`

  This contains an array of process definitions. Each process is connected to the previous one in the sequence through the ports specified in `in_ports` and `out_ports`, if they are specified.

  Each process definitions contain the following keys:

  - `name`: The name to give the process, also used to look up port names.
  - `type`: The type of the process, see `available_processes` for the available options.
  - `out_ports` (optional): a list of port names, which will be connected to the `in_ports` of the next process.

  - `in_ports` (optional): a list of port names, which will be connected to the `out_ports` of the previous process.
  - `parameters` (optional): an object containing parameters specific to this process type (e.g. `path` in the above example).

- `connections` (optional)

  A list of connections to make between the processes, to allow the creation of arbitrary (non-linear) process graphs.

  Each entry should be an array containing two strings representing the output port and input port to connect. Each of these should be of the form `process_name.port_name`.

  For example, this is equivalent to the above configuration:

```
{
  "version": 0,
  "processes": [
    {
      "name": "input",
      "type": "read_adm_bw64",
      "parameters": {
        "path": "/tmp/in.wav"
      }
    },
    {
      "name": "output",
      "type": "write_adm_bw64",
      "parameters": {
        "path": "/tmp/out.wav"
      }
    }
  ],
  "connections": [
    ["input.out_axml", "output.in_axml"],
    ["input.out_samples", "output.in_samples"]
  ]
}
```

## 2.1 Running Processes

To run the processes described by a configuration file, use the `eat-process` tool. This takes a name of a configuration file, and some options.

Options can be used to specify or override a value in the configuration file. For example, the paths in the above examples could be omitted, and a user could run:

```
eat-process example.json -o input.path in.wav -o output.path out.wav
```

Each option name (e.g. `input.path`) is the name of a process, followed by a `.` and an option name for that process. Option names are turned into rfc6901 "JSON pointers", by replacing `.` with `/` and prepending a `/`, so it's possible to modify nested objects and arrays, too.

Setting options on the command-line like this is equivalent to adding or replacing values in the `parameters` block in the configuration file.

There are two forms of this for different situations:

- `-o` or `--option`: The value is parsed as JSON. If this fails, then the value is assumed to be a string.
- `-s` or `--strict-option`: The value is parsed as JSON. If this fails, an error is raised.

`-o` is most useful or interactive usage, while `-s` can help prevent errors when used in scripts, but requires strings to be quoted.

`-p` or `--progress` shows a progress bar while processing.

## 2.2 Available Process Types

The following process types are available:

**`read_adm`**

read ADM data from a BW64 file

**Parameters**

- **`path`** (*`string`*) – path to wav file to read

**Output Ports**

- **`out_axml`** (*`Data<ADMData>`*) – output ADM data

**`read_bw64`**

read samples from a BW64 file

**Parameters**

- **`path`** (*`string`*) – path to wav file to read
- **`block_size`** (*`int`*) – size of chunks to read

**Output Ports**

- **`out_samples`** (*`Stream<InterleavedBlockPtr>`*) – output samples

**`read_adm_bw64`**

read ADM data and samples from a BW64 file

**Parameters**

- **`path`** (*`string`*) – path to wav file to read
- **`block_size`** (*`int`*) – size of chunks to read

**Output Ports**

- **`out_samples`** (*`Stream<InterleavedBlockPtr>`*) – output samples
- **`out_axml`** (*`Data<ADMData>`*) – output ADM data

**`write_adm_bw64`**

write ADM data and samples to a BW64 file

**Parameters**

- **`path`** (*`string`*) – path to wav file to write

**Input Ports**

- **`in_samples`** (*`Stream<InterleavedBlockPtr>`*) – input samples

- **in_axml** (*Data<ADMData>*) – input ADM data

**write_bw64**

write samples to a BW64 file

**Parameters**

- **path** (*string*) – path to wav file to write

**Input Ports**

- **in_samples** (*Stream<InterleavedBlockPtr>*) – input samples

**remove_unused**

remove unreferenced elements from an ADM document, and re-pack the channels to remove unreferenced channels

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data
- **in_samples** (*Stream<InterleavedBlockPtr>*) – input samples

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data
- **out_samples** (*Stream<InterleavedBlockPtr>*) – output samples

**remove_unused_elements**

remove unreferenced elements from an ADM document

in contrast with make_remove_unused, this doesn't do anything with the audio, so can be useful if previous changes will not have affected the use of channels

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

**remove_elements**

remove ADM elements with given IDs

**Parameters**

- **ids** (*array of strings*) – IDs of elements to remove

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

**validate**

Check ADM data against a given profile. Prints any errors and raises an exception if any errors are fined.

**Parameters**

- **profile** (*object*) – Profile specification; `type` specifies the profile type. The following types are defined:

- `itu_emission`: the ITU emission profile. `level` (int from 0 to 2) specifies the profile level.

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

`fix_ds_frequency`

add a `frequency` element with `lowPass="120"` for DirectSpeakers channels with `LFE` in their name

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

`fix_block_durations`

calls `adm::updateBlockFormatDurations()` to fix rounding errors in audioBlockFormat durations

---

**Note:** There is currently no limit to the amount that the durations may be modified by – they are always set to match the rtime of the block after, or to match the end of the object/programme/file.

---

---

**Note:** The length of the audioProgramme is not currently inferred from the file length, so must be specified.

---

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

`fix_stream_pack_refs`

removes audioPackFormatIDRef in audioStreamFormats that are of type PCM and have an audioChannelFormatIDRef

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

`convert_track_stream_to_channel`

Replace audioTrackUid->audioTrackFormat->audioStreamFormat->audioChannelFormat references with audioTrackUid->audioChannelFormat references.

This doesn't remove any unused elements, so use *remove_unused_elements* or *remove_unused* after this.

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

`render`

render ADM to loudspeaker signals according to BS.2127

**Parameters**

- **layout** (*string*) – BS.2051 layout name

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data
- **in_samples** (*Stream<InterleavedBlockPtr>*) – input samples

**Output Ports**

- **out_samples** (*Stream<InterleavedBlockPtr>*) – output samples

`add_block_rtimes`

ensure that blocks with a specified duration have an rtime

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

`measure_loudness`

measure loudness of loudspeaker signals according to BS.1770

**Parameters**

- **layout** (*string*) – BS.2051 layout name

**Input Ports**

- **in_samples** (*Stream<InterleavedBlockPtr>*) – input samples

**Output Ports**

- **out_loudness** (*Data<adm::LoudnessMetadata>*) – output loudness data

`set_programme_loudness`

set audioProgramme loudness metadata

**Parameters**

- **id** (*string*) – audioProgrammeId to modify

**Input Ports**

- **in_loudness** (*Data<adm::LoudnessMetadata>*) – input loudness data
- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

`update_all_programme_loudnesses`

measure the loudness of all audioProgrammes (by rendering them to 4+5+0) and updates the axml to match

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data
- **in_samples** (*Stream<InterleavedBlockPtr>*) – input samples

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

`set_profiles`

set the list of profiles in an ADM document

**Parameters**

- **profiles** (*list*) – see *validate*

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

`set_position_defaults`

add explicit default values for elevation and Z position coordinates

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

`remove_silent_atu`

replace silent audioTrackUID references (with ID 0) with real audioTrackUIDs that reference a silent track

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data
- **in_samples** (*Stream<InterleavedBlockPtr>*) – input samples

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data
- **out_samples** (*Stream<InterleavedBlockPtr>*) – output samples

`remove_jump_position`

Remove the jumpPosition sub-elements from audioBlockFormat sub elements of type objects. Where interpolationLength is set such that the interpolation does not occur across the whole block, split into two blocks representing the interpolated and fixed parts.

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

`resample_blocks`

Change the timing information of audioBlockFromat sub-elements, such that no block is shorter than min-duration. The first block is a special case, in that if it has a duration of 0, that will be preserved regardless of min-duration. The min-duration parameter is in adm time format, eg 100S44100 for fractional representation or 00:00:00050 for timecode representation The representation format used for min-duration must match that used in the audioBlockFormats of the input xml.

**Parameters**

- **min-duration** (*string*) – The minimum duration allowed for output blocks, in adm time format

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

**remove_object_times_data_safe**

remove time/duration from audioObjects where it is safe to do so (doesn't potentially change the rendering) and can be done by only changing the metadata (no audio changes, no converting common definitions audioChannelFormats to real audioChannelFormats

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

**remove_object_times_common_unsafe**

remove start and duration from audioObjects which only reference common definitions audioChannelFormats

this could cause rendering changes if there are non-zero samples outside the range of the audioObject, but should be safe on EPS output

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

**remove_importance**

remove importance values from all audioObjects, audioPackFormats and audioBlockFormats

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

**drop_blockformat_subelements**

Drop specified sub-elements from all AudioBlockFormats This processor simply removes the subelements and does not attempt to replace them in any way.

**Parameters**

- **objects_subelements** (*list*) – A list of subelements to remove from AudioBlockFormats with Object type. Valid values are "Diffuse", "ChannelLock", "ObjectDivergence", "JumpPosition", "ScreenRef", "Width", "Depth", "Height", "Gain", "Importance", "Headlocked" and "HeadphoneVirtualise"

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data
- **out_axml** – output ADM data

**rewrite_content_objects_emission**

rewrite the programme-content-object structure to make it compatible with emission profile rules

the restrictions are:

- audioContents can only reference one audioObject
- only audioObjects containing Objects content can be nested
- there's an undefined "maximum nest level" of 2

this may drop audioContents or audioObjects that are too nested (only ones that have audioObject references), so any information which applies to the audioObjects below will be lost

**Parameters**

- **max_objects_depth=2** (*int*) – the maximum object depth allowed for any object, defined as the maximum number of audioObject references between the object and any objects with audio content (audioPackFormat/audioTrackFormat)

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

**infer_object_interact**

ensure that all audioObjects have an interact parameter based on the presence or absence of the audioObjectInteraction element

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

**set_version**

set the audioFormatExtended version

**Input Ports**

- **in_axml** (*Data<ADMData>*) – input ADM data

**Output Ports**

- **out_axml** (*Data<ADMData>*) – output ADM data

**Parameters**

- **version** (*string*) – the version string to set

**set_content_dialogue_default**

set missing audioContent dialogue values to mixed

**Input Ports**

- **in_axml** (*data<admdata>*) – input adm data

**Output Ports**

- **out_axml** (*data<admdata>*) – output adm data

CHAPTER

# THREE

# LIBRARY API

## 3.1 File Hierarchy

## 3.2 Full API

### 3.2.1 Namespaces

#### Namespace adm

#### Namespace eat

**Contents**



#### Namespaces

### Namespace eat::admx

**Contents**



### Functions



### Namespace eat::framework

**Contents**



### Classes

- *Template Class NullSink*
- *Class Plan*
- *Class Port*
- *Class Process*
- *Class StreamingAtomicProcess*
- *Template Class StreamPort*
- *Class StreamPortBase*
- *Class ValidationError*
- *Template Class ValuePtr*

### Functions

- *Function eat::framework::always_assert*
- *Template Function eat::framework::copy_shared_ptr*
- *Specialized Template Function eat::framework::copy_shared_ptr< adm::Document >*
- *Function eat::framework::evaluate*
- *Function eat::framework::flatten*
- *Function eat::framework::plan*
- *Function eat::framework::run_with_progress*
- *Function eat::framework::validate*

### Typedefs

- *Typedef eat::framework::AtomicProcessPtr*
- *Typedef eat::framework::CompositeProcessPtr*
- *Typedef eat::framework::DataPortBasePtr*
- *Typedef eat::framework::DataPortPtr*
- *Typedef eat::framework::ExecStepPtr*
- *Typedef eat::framework::FunctionalAtomicProcessPtr*
- *Typedef eat::framework::GraphPtr*
- *Typedef eat::framework::PortPtr*
- *Typedef eat::framework::ProcessPtr*
- *Typedef eat::framework::StreamingAtomicProcessPtr*
- *Typedef eat::framework::StreamPortBasePtr*
- *Typedef eat::framework::StreamPortPtr*

## Namespace eat::process

**Contents**



## Namespaces



## Classes

- *Class TempDir*

### Enums

- *Enum LanguageCodeType*

### Functions

- *Function eat::process::clear_id*
- *Function eat::process::de_duplicate_zero_length_blocks*
- *Function eat::process::format_language_code_types*
- *Function eat::process::load_chna*
- *Function eat::process::make_add_block_rtimes*
- *Function eat::process::make_apply_channel_mapping*
- *Function eat::process::make_block_resampler*
- *Function eat::process::make_block_subelement_dropper*
- *Function eat::process::make_chna*
- *Function eat::process::make_convert_track_stream_to_channel*
- *Function eat::process::make_fix_block_durations*
- *Function eat::process::make_fix_ds_frequency*
- *Function eat::process::make_fix_stream_pack_refs*
- *Function eat::process::make_infer_object_interact*
- *Function eat::process::make_jump_position_remover*
- *Function eat::process::make_measure_loudness*
- *Function eat::process::make_read_adm*
- *Function eat::process::make_read_adm_bw64*
- *Function eat::process::make_read_bw64*
- *Function eat::process::make_remove_elements*
- *Function eat::process::make_remove_importance*
- *Function eat::process::make_remove_object_times_common_unsafe*
- *Function eat::process::make_remove_object_times_data_safe*
- *Function eat::process::make_remove_silent_atu*
- *Function eat::process::make_remove_unused*
- *Function eat::process::make_remove_unused_elements*
- *Function eat::process::make_rewrite_content_objects_emission*
- *Function eat::process::make_set_content_dialogue_default*
- *Function eat::process::make_set_position_defaults*

- *Function eat::process::make_set_profiles*
- *Function eat::process::make_set_programme_loudness*
- *Function eat::process::make_set_version*
- *Function eat::process::make_update_all_programme_loudnesses*
- *Function eat::process::make_validate*
- *Function eat::process::make_write_adm_bw64*
- *Function eat::process::make_write_bw64*
- *Function eat::process::only_object_type*
- *Function eat::process::parse_droppable*
- *Function eat::process::parse_language_code*
- *Function eat::process::referenced_channel_formats*
- *Function eat::process::remove_jump_position*
- *Function eat::process::resample_to_minimum_preserving_zero*
- *Function eat::process::split*
- *Function eat::process::validate_config*

### Typedefs

- *Typedef eat::process::channel_map_t*
- *Typedef eat::process::ChannelMapping*
- *Typedef eat::process::ElementIds*
- *Typedef eat::process::InterleavedBlockPtr*
- *Typedef eat::process::PlanarBlockPtr*

### Variables

- *Variable eat::process::language_codes*

## Namespace eat::process::profiles

**Contents**

### Classes

- *Struct ITUEmissionProfile*

### Typedefs

- *Typedef eat::process::profiles::Profile*

### Namespace eat::process::validation

**Contents**



### Detailed Description

This implements validation for ADM documents.

To use this as a process, see `eat::process::make_validate()`.

To use it directly, see `make_profile_validator()` and `make_emission_profile_validator()` to make a validator, `ProfileValidator::run()` to run it, and `any_messages()` and `format_results()` to interpret the results.

In general, validation checks are implemented as a struct whose members configure the check, and which has a run function which performs the check

Run yields a vector of message objects, and the type of these corresponds with the type of the check. For example, the `NumElements` check results in `NumElementsMessage` messages.

These are combined together with variants, so a `Check` is one of any known checks, and a `Message` is one of any known message.

Both checks and message types have additional functions to enable meta-programming:

- A static name method gives the name of the check or message (the same as the class name)
- A visit function accepts a callback and calls it once for each member with the name of the member and a reference to the member.

These can be used to serialise and de-serialise checks and messages to JSON, for example.

### Classes

- *Template Struct ElementInList*
- *Template Struct ElementInListMessage*
- *Template Struct ElementInRange*
- *Template Struct ElementInRangeMessage*
- *Struct ElementPresent*
- *Struct ElementPresentMessage*
- *Struct NumElements*
- *Struct NumElementsMessage*
- *Struct ObjectContentOrNested*
- *Struct ObjectContentOrNestedMessage*
- *Template Struct Range*
- *Struct StringLength*
- *Struct StringLengthMessage*
- *Template Struct UniqueElements*
- *Template Struct UniqueElementsMessage*
- *Struct ValidationResult*
- *Struct ValidLanguage*
- *Struct ValidLanguageMessage*
- *Class ProfileValidator*

### Functions

- *Function eat::process::validation::any_messages*
- *Function eat::process::validation::format_check*
- *Function eat::process::validation::format_message*
- *Function eat::process::validation::format_results*
- *Function eat::process::validation::make_emission_profile_validator*
- *Function eat::process::validation::make_profile_validator*

### Typedefs

- *Typedef eat::process::validation::Check*
- *Typedef eat::process::validation::CountRange*
- *Typedef eat::process::validation::Message*
- *Typedef eat::process::validation::ValidationResults*

## Namespace eat::render

**Contents**



### Classes

- *Struct ADMPath*
- *Struct DefaultStart*
- *Struct DirectSpeakersRenderingItem*
- *Struct DirectTrackSpec*
- *Struct HOARenderingItem*
- *Struct HOATypeMetadata*
- *Struct MonoRenderingItem*
- *Struct ObjectRenderingItem*
- *Struct RenderingItem*
- *Struct SelectionOptions*
- *Struct SelectionOptionsId*
- *Struct SelectionResult*
- *Struct SilentTrackSpec*
- *Class ItemSelectionError*

### Functions

- *Function eat::render::make_render*
- *Function eat::render::select_items*
- *Function eat::render::selection_options_from_ids*

### Typedefs

- *Typedef eat::render::ContentIdStart*
- *Typedef eat::render::ContentStart*
- *Typedef eat::render::ObjectIdStart*
- *Typedef eat::render::ObjectStart*
- *Typedef eat::render::ProgrammeIdStart*
- *Typedef eat::render::ProgrammeStart*
- *Typedef eat::render::SelectionStart*
- *Typedef eat::render::SelectionStartId*
- *Typedef eat::render::TrackSpec*

## Namespace eat::testing

**Contents**



### Classes

- *Class TempDir*

### Functions

- *Function eat::testing::files_equal*

### Namespace eat::utilities

**Contents**



#### Namespaces



#### Classes



#### Functions



#### Typedefs



### Namespace eat::utilities::element_visitor

**Contents**

### Classes

- *Class Visitable*

### Functions

- *Function eat::utilities::element_visitor::dotted_path*
- *Function eat::utilities::element_visitor::format_path*
- *Function eat::utilities::element_visitor::path_to_strings*
- *Function eat::utilities::element_visitor::visit(const VisitablePtr&, const std::vector<std::string>&, const std::function<void(const Path&path)>&)*
- *Function eat::utilities::element_visitor::visit(std::shared_ptr<adm::Document>, const std::vector<std::string>&, const std::function<void(const Path&path)>&)*

### Typedefs

- *Typedef eat::utilities::element_visitor::Path*
- *Typedef eat::utilities::element_visitor::VisitablePtr*

## 3.2.2 Classes and Structs

### Template Struct MakeBuffer

- Defined in file_include_eat_framework_process.hpp

### Struct Documentation

template<typename **T**>

struct **MakeBuffer**

for specialising get_buffer_writer / get_buffer_reader for different types

#### Public Functions

template<>
*ProcessPtr* **get_buffer_reader**(const std::string &name)

template<>
*ProcessPtr* **get_buffer_writer**(const std::string &name)

#### Public Static Functions

static *ProcessPtr* **get_buffer_writer**(const std::string &name)

static *ProcessPtr* **get_buffer_reader**(const std::string &name)

### Struct ADMData

- Defined in file_include_eat_process_adm_bw64.hpp

### Struct Documentation

struct **ADMData**

stores ADM information associated with some stream of audio

CHNA information is merged into document, with the channel number for each audioChannelUID stored in channel_map

#### Public Members

framework::*ValuePtr*<adm::Document> **document**

*channel_map_t* **channel_map**

### Struct AudioInterval

- Defined in file_include_eat_process_silence_detect.hpp

### Struct Documentation

struct **AudioInterval**

#### Public Members

std::size_t **start** = {0}

std::size_t **length** = {0}

### Struct BlockDescription

- Defined in file_include_eat_process_block.hpp

### Struct Documentation

struct **BlockDescription**

description of a block of samples (independent of storage format)

#### Public Members

std::size_t **sample_count**

number of samples

std::size_t **channel_count**

number of channels

unsigned int **sample_rate**

sample rate in Hz

### Struct CartesianPosition

- Defined in file_include_eat_process_directspeaker_conversion.hpp

### Struct Documentation

struct **CartesianPosition**

#### Public Members

float **x**

float **y**

float **z**

### Struct Constraint

- Defined in file_include_eat_process_limit_interaction.hpp

### Struct Documentation

struct **Constraint**

#### Public Members

float **min** = {std::numeric_limits<float>::min()}

float **max** = {std::numeric_limits<float>::max()}

### Struct GainInteractionConstraint

- Defined in file_include_eat_process_limit_interaction.hpp

### Struct Documentation

struct **GainInteractionConstraint**

#### Public Members

std::optional<*Constraint*> **min**

std::optional<*Constraint*> **max**

bool **permitted** = {true}

### Struct InteractionLimiter::Config

- Defined in file_include_eat_process_limit_interaction.hpp

### Nested Relationships

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

### Struct Documentation

struct **Config**

#### Public Members

bool **remove_disabled_ranges** = {false}

std::optional<*GainInteractionConstraint*> **gain_range**

std::optional<*PositionInteractionConstraint*> **position_range**

std::vector<*Droppable*> **types_to_disable**

### Struct PositionConstraint

- Defined in file_include_eat_process_limit_interaction.hpp

### Struct Documentation

struct **PositionConstraint**

#### Public Members

std::optional<*Constraint*> **min**

std::optional<*Constraint*> **max**

bool **permitted** = {true}

### Struct PositionInteractionConstraint

- Defined in file_include_eat_process_limit_interaction.hpp

### Struct Documentation

struct **PositionInteractionConstraint**

#### Public Members

*PositionConstraint* **azimuth**

*PositionConstraint* **elevation**

*PositionConstraint* **distance**

*PositionConstraint* **x**

*PositionConstraint* **y**

*PositionConstraint* **z**

### Struct ITUEmissionProfile

- Defined in file_include_eat_process_profiles.hpp

### Struct Documentation

struct **ITUEmissionProfile**

#### Public Functions

inline **ITUEmissionProfile**(int level)

inline int **level**() const

### Struct SilenceDetectionConfig

- Defined in file_include_eat_process_silence_detect.hpp

### Struct Documentation

struct **SilenceDetectionConfig**

#### Public Members

std::size_t **minimum_length** = {10}

Number of consecutive samples that must test below the threshold to register as silence

double **threshold** = {-95.0}

Value in dB that will be used as a threshold for detecting silence

### Template Struct ElementInList

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

template<typename **T**>

struct **ElementInList**

check that an element is one of a given list of values

T is the unwrapped element type, e.g. string

#### Public Types

using **Message** = *ElementInListMessage*<*T*>

#### Public Functions

inline std::vector<*Message*> **run**(const *ADMData* &adm) const

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

path to the elements to check

std::vector<*T*> **options**

acceptable options

#### Public Static Functions

static inline std::string **name**()

### Template Struct ElementInListMessage

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

template<typename **T**>

struct **ElementInListMessage**

#### Public Functions

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

*T* **value**

#### Public Static Functions

static inline std::string **name**()

### Template Struct ElementInRange

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

template<typename **T**>

struct **ElementInRange**

check that an element is in the given range

T is the unwrapped element type (e.g. float or int)

#### Public Types

using **Message** = *ElementInRangeMessage*<*T*>

#### Public Functions

inline std::vector<*Message*> **run**(const *ADMData* &adm) const

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

*Range*<*T*> **range**

#### Public Static Functions

static inline std::string **name**()

### Template Struct ElementInRangeMessage

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

template<typename **T**>

struct **ElementInRangeMessage**

#### Public Functions

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

*T* **value**

#### Public Static Functions

static inline std::string **name**()

### Struct ElementPresent

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **ElementPresent**

check if an element is present

this is really the same as *NumElements*, but with different formatting

#### Public Types

using **Message** = *ElementPresentMessage*

#### Public Functions

std::vector<*Message*> **run**(const *ADMData* &adm) const

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

std::string **element**

bool **present**

#### Public Static Functions

static inline std::string **name**()

### Struct ElementPresentMessage

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **ElementPresentMessage**

#### Public Functions

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

std::string **element**

bool **present**

#### Public Static Functions

static inline std::string **name**()

### Struct NumElements

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **NumElements**

generic check for the number of elements within a containing element

this can check top-level elements, references, or real sub-elements

#### Public Types

using **Message** = *NumElementsMessage*

#### Public Functions

std::vector<*Message*> **run**(const *ADMData* &adm) const

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

path to the containing element (empty to check top-level elements

std::string **element**

name of the element to count

*CountRange* **range**

acceptable number of elements

std::string **relationship** = "elements"

plural relationship type only used for formatting the check/message

#### Public Static Functions

static inline std::string **name**()

### Struct NumElementsMessage

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **NumElementsMessage**

#### Public Functions

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

std::string **element**

size_t **n**

std::string **relationship** = "elements"

#### Public Static Functions

static inline std::string **name**()

### Struct ObjectContentOrNested

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **ObjectContentOrNested**

check that audioObjects have either audio content (pack or trackUid refs) or nested audioObjects, not both or neither

#### Public Types

using **Message** = *ObjectContentOrNestedMessage*

#### Public Functions

std::vector<*Message*> **run**(const *ADMData* &adm) const

template<typename **F**>
inline void **visit**(*F*)

#### Public Static Functions

static inline std::string **name**()

### Struct ObjectContentOrNestedMessage

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **ObjectContentOrNestedMessage**

#### Public Functions

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

adm::AudioObjectId **object_id**

bool **both**

#### Public Static Functions

static inline std::string **name**()

### Template Struct Range

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

template<typename **T**>

struct **Range**

a range check for numbers

#### Public Functions

inline bool **operator()**(*T* n) const

inline std::string **format**() const

#### Public Members

std::optional<*T*> **lower_limit**

std::optional<*T*> **upper_limit**

#### Public Static Functions

static inline *Range* **up_to**(*T* upper_limit)

static inline *Range* **at_least**(*T* lower_limit)

static inline *Range* **between**(*T* lower_limit, *T* upper_limit)

static inline *Range* **exactly**(*T* limit)

### Struct StringLength

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **StringLength**

check the length of a string

#### Public Types

using **Message** = *StringLengthMessage*

#### Public Functions

std::vector<*Message*> **run**(const *ADMData* &adm) const

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

path to a string

*CountRange* **range**

acceptable number of characters

#### Public Static Functions

static inline std::string **name**()

### Struct StringLengthMessage

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **StringLengthMessage**

#### Public Functions

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

size_t **n**

#### Public Static Functions

static inline std::string **name**()

### Template Struct UniqueElements

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

template<typename **T**>

struct **UniqueElements**

check elements for uniqueness

for each element visited by path1, the elements from that visited by path2 must be unique

T is the type of the unwrapped element to check, e.g. string

#### Public Types

using **Message** = *UniqueElementsMessage*<*T*>

#### Public Functions

inline std::vector<*Message*> **run**(const *ADMData* &adm) const

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path1**

path to the element which contains the unique values

std::vector<std::string> **path2**

path from the elements visited by path1 to the elements to check

#### Public Static Functions

static inline std::string **name**()

### Template Struct UniqueElementsMessage

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

template<typename **T**>

struct **UniqueElementsMessage**

#### Public Functions

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path1**

*T* **value**

std::vector<std::string> **path2a**

std::vector<std::string> **path2b**

#### Public Static Functions

static inline std::string **name**()

### Struct ValidationResult

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **ValidationResult**

result of running a single check; holds the check, and the messages that it resulted in

#### Public Members

*Check* **check**

std::vector<*Message*> **messages**

### Struct ValidLanguage

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **ValidLanguage**

check that an element contains a valid language code

#### Public Types

using **Message** = *ValidLanguageMessage*

#### Public Functions

std::vector<*Message*> **run**(const *ADMData* &adm) const

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

path to a string containing the language code

*LanguageCodeType* **type**

acceptable language code types

#### Public Static Functions

static inline std::string **name**()

### Struct ValidLanguageMessage

- Defined in file_include_eat_process_validate.hpp

### Struct Documentation

struct **ValidLanguageMessage**

#### Public Functions

template<typename **F**>
inline void **visit**(*F* f)

#### Public Members

std::vector<std::string> **path**

std::string **value**

#### Public Static Functions

static inline std::string **name**()

### Struct ADMPath

- Defined in file_include_eat_render_rendering_items.hpp

### Struct Documentation

struct **ADMPath**

#### Public Members

std::shared_ptr<adm::AudioProgramme> **audioProgramme**

std::shared_ptr<adm::AudioContent> **audioContent**

std::vector<std::shared_ptr<adm::AudioObject>> **audioObjects**

std::vector<std::shared_ptr<adm::AudioPackFormat>> **audioPackFormats**

std::shared_ptr<adm::AudioChannelFormat> **audioChannelFormat**

### Struct DefaultStart

- Defined in file_include_eat_render_rendering_items.hpp

### Struct Documentation

struct **DefaultStart**

### Struct DirectSpeakersRenderingItem

- Defined in file_include_eat_render_rendering_items.hpp

### Inheritance Relationships

### Base Type

- `public eat::render::MonoRenderingItem` (*Struct MonoRenderingItem*)

### Struct Documentation

struct **DirectSpeakersRenderingItem** : public eat::render::*MonoRenderingItem*

### Struct DirectTrackSpec

- Defined in file_include_eat_render_rendering_items.hpp

### Struct Documentation

struct **DirectTrackSpec**

#### Public Functions

**auto operator (const DirectTrackSpec &) const =default**

#### Public Members

std::shared_ptr<adm::AudioTrackUid> **track**

### Struct HOARenderingItem

- Defined in file_include_eat_render_rendering_items.hpp

### Inheritance Relationships

### Base Type

- `public eat::render::RenderingItem` (*Struct RenderingItem*)

### Struct Documentation

struct **HOARenderingItem** : public eat::render::*RenderingItem*

#### Public Members

std::vector<*TrackSpec*> **tracks**

std::vector<*ADMPath*> **adm_paths**

std::vector<*HOATypeMetadata*> **type_metadata**

### Struct HOATypeMetadata

- Defined in file_include_eat_render_rendering_items.hpp

### Struct Documentation

struct **HOATypeMetadata**

#### Public Members

std::optional<adm::Time> **rtime**

std::optional<adm::Time> **duration**

std::vector<int> **orders**

std::vector<int> **degrees**

std::string **normalization** = "SN3D"

std::optional<double> **nfcRefDist** = {}

bool **screenRef** = false

### Struct MonoRenderingItem

- Defined in file_include_eat_render_rendering_items.hpp

### Inheritance Relationships

### Base Type

- `public eat::render::RenderingItem` (*Struct RenderingItem*)

### Derived Types

- `public eat::render::DirectSpeakersRenderingItem` (*Struct DirectSpeakersRenderingItem*)
- `public eat::render::ObjectRenderingItem` (*Struct ObjectRenderingItem*)

### Struct Documentation

struct **MonoRenderingItem** : public eat::render::*RenderingItem*

Subclassed by *eat::render::DirectSpeakersRenderingItem*, *eat::render::ObjectRenderingItem*

#### Public Members

*TrackSpec* **track_spec**

*ADMPath* **adm_path**

### Struct ObjectRenderingItem

- Defined in file_include_eat_render_rendering_items.hpp

### Inheritance Relationships

### Base Type

- `public eat::render::MonoRenderingItem` (*Struct MonoRenderingItem*)

### Struct Documentation

struct **ObjectRenderingItem** : public eat::render::*MonoRenderingItem*

### Struct RenderingItem

- Defined in file_include_eat_render_rendering_items.hpp

### Inheritance Relationships

### Derived Types

- `public eat::render::HOARenderingItem` (*Struct HOARenderingItem*)
- `public eat::render::MonoRenderingItem` (*Struct MonoRenderingItem*)

### Struct Documentation

struct **RenderingItem**

Subclassed by *eat::render::HOARenderingItem*, *eat::render::MonoRenderingItem*

#### Public Functions

virtual **~RenderingItem**() = default

### Struct SelectionOptions

- Defined in file_include_eat_render_rendering_items.hpp

### Struct Documentation

struct **SelectionOptions**

#### Public Functions

**SelectionOptions**() = default

**SelectionOptions**(*SelectionStart* start)

#### Public Members

*SelectionStart* **start** = *DefaultStart*{}

### Struct SelectionOptionsId

- Defined in file_include_eat_render_rendering_items_options_by_id.hpp

### Struct Documentation

struct **SelectionOptionsId**

#### Public Functions

**SelectionOptionsId**() = default

**SelectionOptionsId**(*SelectionStartId* start_)

#### Public Members

*SelectionStartId* **start** = *DefaultStart*{}

### Struct SelectionResult

- Defined in file_include_eat_render_rendering_items.hpp

### Struct Documentation

struct **SelectionResult**

#### Public Members

std::vector<std::shared_ptr<*RenderingItem*>> **items**

std::vector<std::string> **warnings**

### Struct SilentTrackSpec

- Defined in file_include_eat_render_rendering_items.hpp

### Struct Documentation

struct **SilentTrackSpec**

#### Public Functions

**auto operator (const SilentTrackSpec &) const =default**

### Template Struct ForEachElement

- Defined in file_include_eat_utilities_for_each_element.hpp

### Struct Documentation

template<template<typename Element> typename **T**>

struct **ForEachElement**

borrowed from https://github.com/ebu/libadm/pull/155

store a value for each top-level element type, with access by type

the value is given by the template T, so that *get<AudioProgramme>()* returns a T<AudioProgramme>, for example

#### Public Functions

template<typename **El**>
inline *T*<*El*> &**get**()

get one of the stored values

template<typename **F**>
inline void **visit**(*F* f)

call f on each of the stored values

### Class AssertionError

- Defined in file_include_eat_framework_exceptions.hpp

### Inheritance Relationships

### Base Type

- `public runtime_error`

### Class Documentation

class **AssertionError** : public runtime_error

### Class AtomicProcess

- Defined in file_include_eat_framework_process.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::Process` (*Class Process*)

### Derived Types

- `public eat::framework::FunctionalAtomicProcess` (*Class FunctionalAtomicProcess*)
- `public eat::framework::StreamingAtomicProcess` (*Class StreamingAtomicProcess*)

### Class Documentation

class **AtomicProcess** : public eat::framework::*Process*

abstract process which actually does something (*FunctionalAtomicProcess* or *StreamingAtomicProcess*), rather than a graph of other processes (*CompositeProcess*)

Subclassed by *eat::framework::FunctionalAtomicProcess*, *eat::framework::StreamingAtomicProcess*

### Class CompositeProcess

- Defined in file_include_eat_framework_process.hpp

### Inheritance Relationships

### Base Types

- `public eat::framework::Process` (*Class Process*)
- `public eat::framework::Graph` (*Class Graph*)

### Class Documentation

class **CompositeProcess** : public eat::framework::*Process*, public eat::framework::*Graph*

A process which just contains some other processes, and connections between them

#### Public Functions

void **connect**(const *PortPtr* &a, const *PortPtr* &b)

### Template Class DataPort

- Defined in file_include_eat_framework_process.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::DataPortBase` (*Class DataPortBase*)

### Class Documentation

template<typename **T**>

class **DataPort** : public eat::framework::*DataPortBase*

non-streaming data port for a specific type, T

T should be copyable and movable (value semantics), so that processes will not interact (i.e. see changes from non-upstream processes), and don't have to copy manually

for heavy types (e.g. vector) it may be more efficient to use a smart pointer with const contents (like std::shared_ptr<const std::vector<...>>) to avoid copying unnecessarily

for types with RAII behaviour, readers should move from get_value to ensure that resources are freed as soon as possible

#### Public Functions

virtual bool **compatible**(const *PortPtr* &other) const override

are connections between this and other valid (i.e. the same type)

template<typename **U**>
void **set_value**(*U* &&value)

set the value — use this from a process for which this port is an output

const *T* &**get_value**() const

get the value — use this from a process for which this port is an input

*T* &**get_value**()

virtual void **move_to**(*DataPortBase* &other) override

virtual void **copy_to**(*DataPortBase* &other) override

### Class DataPortBase

- Defined in file_include_eat_framework_process.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::Port` (*Class Port*)

### Derived Type

- `public eat::framework::DataPort< T >` (*Template Class DataPort*)

### Class Documentation

class **DataPortBase** : public eat::framework::*Port*

abstract port for non-streaming data; see DataPort<T>

Subclassed by *eat::framework::DataPort< T >*

#### Public Functions

virtual void **move_to**(*DataPortBase*&) = 0

virtual void **copy_to**(*DataPortBase*&) = 0

### Template Class DataSink

- Defined in file_include_eat_framework_utility_processes.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::FunctionalAtomicProcess` (*Class FunctionalAtomicProcess*)

### Class Documentation

template<typename **T**>

class **DataSink** : public eat::framework::*FunctionalAtomicProcess*

process with an input port whose value is saved

ports:

- in (`DataPort`<T>) : input data, accessible with get_value

#### Public Functions

inline **DataSink**(const std::string &name)

inline *T* &**get_value**()

inline virtual void **process**() override

### Template Class DataSource

- Defined in file_include_eat_framework_utility_processes.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::FunctionalAtomicProcess` (*Class FunctionalAtomicProcess*)

### Class Documentation

template<typename **T**>

class **DataSource** : public eat::framework::*FunctionalAtomicProcess*

process with an output port that is set to a value provided in the constructor or set_value

ports:

- out (*DataPort*<T>) : output data

#### Public Functions

inline **DataSource**(const std::string &name, *T* value)

inline explicit **DataSource**(const std::string &name)

inline void **set_value**(*T* value)

inline virtual void **process**() override

### Class ExecStep

- Defined in file_include_eat_framework_evaluate.hpp

### Class Documentation

class **ExecStep**

representation of a step within an execution plan

this doesn't do much more than std::function, but this is where we would add progress callbacks

#### Public Functions

inline virtual **~ExecStep**()

virtual void **run**() = 0

virtual std::string **description**() = 0

## Class FunctionalAtomicProcess

- Defined in file_include_eat_framework_process.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::AtomicProcess` (*Class AtomicProcess*)

### Derived Types

- `public eat::framework::DataSink< T >` (*Template Class DataSink*)
- `public eat::framework::DataSource< T >` (*Template Class DataSource*)
- `public eat::framework::NullSink< T >` (*Template Class NullSink*)
- `public eat::process::BlockResampler` (*Class BlockResampler*)
- `public eat::process::BlockSubElementDropper` (*Class BlockSubElementDropper*)
- `public eat::process::InteractionLimiter` (*Class InteractionLimiter*)
- `public eat::process::JumpPositionRemover` (*Class JumpPositionRemover*)

### Class Documentation

class **FunctionalAtomicProcess** : public eat::framework::*AtomicProcess*

non-streaming process

once all processes connected to input ports have been processed, process() will be called once, before processes connected to the output ports are ran

Subclassed by *eat::framework::DataSink< T >*, *eat::framework::DataSource< T >*, *eat::framework::NullSink< T >*, *eat::process::BlockResampler*, *eat::process::BlockSubElementDropper*, *eat::process::InteractionLimiter*, *eat::process::JumpPositionRemover*

### Public Functions

virtual void **process**() = 0

## Class Graph

- Defined in file_include_eat_framework_process.hpp

## Inheritance Relationships

## Derived Type

- `public eat::framework::CompositeProcess` (*Class CompositeProcess*)

## Class Documentation

class **Graph**

A graph of processes, storing a collection of process references, and connections between the ports

Subclassed by *eat::framework::CompositeProcess*

### Public Functions

inline virtual **~Graph**()

template<typename **T**, typename ...**Args**>
std::shared_ptr<*T*> **add_process**(*Args*&&... args)

construct and register a process with a given type

*ProcessPtr* **register_process**(*ProcessPtr* process)

void **connect**(const *PortPtr* &a, const *PortPtr* &b)

inline const std::vector<*ProcessPtr*> &**get_processes**() const

inline const std::map<*PortPtr*, *PortPtr*> &**get_port_inputs**() const

### Protected Attributes

std::map<*PortPtr*, *PortPtr*> **port_inputs**

### Template Class NullSink

- Defined in file_include_eat_framework_utility_processes.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::FunctionalAtomicProcess` (*Class FunctionalAtomicProcess*)

### Class Documentation

template<typename **T**>

class **NullSink** : public eat::framework::*FunctionalAtomicProcess*

process with an input port whose value is discarded

ports:

- in (*DataPort*<T>) : input data to discard

#### Public Functions

inline **NullSink**(const std::string &name)

inline virtual void **process**() override

### Class Plan

- Defined in file_include_eat_framework_evaluate.hpp

### Class Documentation

class **Plan**

a plan for evaluating a graph

#### Public Functions

inline **Plan**(*Graph* graph, std::vector<*ExecStepPtr*> steps)

inline const *Graph* &**graph**() const

get the actual graph that will be evaluated

this is useful for debugging to see the changes made by the planner

inline const std::vector<*ExecStepPtr*> &**steps**() const

get the steps in the plan

inline void **run**()

run all steps in the plan

### Class Port

- Defined in file_include_eat_framework_process.hpp

### Inheritance Relationships

### Derived Types

- `public eat::framework::DataPortBase` (*Class DataPortBase*)
- `public eat::framework::StreamPortBase` (*Class StreamPortBase*)

### Class Documentation

class **Port**

a port which carries data between processes

whether it's an input or output port depends on how it's connected

Subclassed by *eat::framework::DataPortBase*, *eat::framework::StreamPortBase*

#### Public Functions

explicit **Port**(const std::string &name)

virtual **~Port**() = default

inline const std::string &**name**() const

virtual bool **compatible**(const *PortPtr* &other) const = 0

are connections between this and other valid (i.e. the same type)

### Class Process

- Defined in file_include_eat_framework_process.hpp

### Inheritance Relationships

### Derived Types

- `public eat::framework::AtomicProcess` (*Class AtomicProcess*)
- `public eat::framework::CompositeProcess` (*Class CompositeProcess*)

### Class Documentation

class **Process**

abstract process, for referencing either an atomic or composite process

Subclassed by *eat::framework::AtomicProcess*, *eat::framework::CompositeProcess*

#### Public Functions

explicit **Process**(const std::string &name)

inline virtual **~Process**()

template<typename **T**>
std::shared_ptr<*T*> **add_in_port**(const std::string &name)

construct and register a port with a given type and name

template<typename **T**>
std::shared_ptr<*T*> **add_out_port**(const std::string &name)

construct and register a port with a given type and name

template<typename **T** = *Port*>
std::shared_ptr<*T*> **get_in_port**(const std::string &name) const

get an input port with a given name and type; will throw if there is no port with the given name, or it is not castable to the right type

template<typename **T** = *Port*>
std::shared_ptr<*T*> **get_out_port**(const std::string &name) const

see get_in_port

std::map<std::string, *PortPtr*> **get_port_map**() const

inline const std::map<std::string, *PortPtr*> &**get_in_port_map**() const

inline const std::map<std::string, *PortPtr*> &**get_out_port_map**() const

inline const std::string &**name**() const

### Class StreamingAtomicProcess

- Defined in file_include_eat_framework_process.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::AtomicProcess` (*Class AtomicProcess*)

### Derived Types

- public eat::framework::detail::InMemBufferRead< T >
- public eat::framework::detail::InMemBufferWrite< T >
- public eat::process::InterleavedStreamingAudioSink (*Class InterleavedStreamingAudioSink*)
- public eat::process::InterleavedStreamingAudioSource (*Class InterleavedStreamingAudioSource*)
- public eat::process::SilenceDetector (*Class SilenceDetector*)

### Class Documentation

class **StreamingAtomicProcess** : public eat::framework::*AtomicProcess*

streaming process with the following callbacks:

- initialise() will be called once, after all processes connected to this via non-streaming input ports have been ran (so non-streaming inputs are available)
- process() will be called many times, as long as any streaming ports are not closed; it should read from streaming input ports, and write to streaming output ports, closing them once there's no more data to write
- finalise() will be called once, before processes connected to this via non-streaming output ports are ran

Subclassed by eat::framework::detail::InMemBufferRead< T >, eat::framework::detail::InMemBufferWrite< T >, *eat::process::InterleavedStreamingAudioSink*, *eat::process::InterleavedStreamingAudioSource*, *eat::process::SilenceDetector*

#### Public Functions

inline virtual void **initialise**()

inline virtual void **process**()

inline virtual void **finalise**()

inline virtual std::optional<float> **get_progress**()

get progress for this process as a fraction between 0 and 1 if known

### Template Class StreamPort

- Defined in file_include_eat_framework_process.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::StreamPortBase` (*Class StreamPortBase*)

### Class Documentation

template<typename **T**>

class **StreamPort** : public eat::framework::*StreamPortBase*

stream port containing items of type T

#### Public Functions

virtual bool **compatible**(const *PortPtr* &other) const override

are connections between this and other valid (i.e. the same type)

void **push**(*T* value)

bool **available**() const

*T* **pop**()

virtual void **close**() override

end the stream of data

virtual bool **eof**() override

has the stream ended? true when there's no more data to read, and the other side has called *close()*

virtual bool **eof_triggered**() override

has close been called (i.e. *eof()* will become true once the queue is drained)

virtual void **copy_to**(*StreamPortBase* &other) override

virtual void **move_to**(*StreamPortBase* &other) override

virtual void **clear**() override

virtual *ProcessPtr* **get_buffer_writer**(const std::string &name) override

get a process with a compatible streaming input and a non-streaming output compatible with *get_buffer_writer()*, which writes the inputs to a buffer

virtual *ProcessPtr* **get_buffer_reader**(const std::string &name) override

get a process with a compatible streaming output and a non-streaming input compatible with *get_buffer_reader()*, which reads from a buffer

### Class StreamPortBase

- Defined in file_include_eat_framework_process.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::Port` (*Class Port*)

### Derived Type

- `public eat::framework::StreamPort< T >` (*Template Class StreamPort*)

### Class Documentation

class **StreamPortBase** : public eat::framework::*Port*

port that has a stream of data with a given type (see StreamPort<T>)

the output side calls push(data) n times then *close()* once

the input side calls pop() while available(), and can know that no more data will become available if *eof()*

Subclassed by *eat::framework::StreamPort< T >*

#### Public Functions

virtual void **close**() = 0

end the stream of data

virtual bool **eof**() = 0

has the stream ended? true when there's no more data to read, and the other side has called *close()*

virtual bool **eof_triggered**() = 0

has close been called (i.e. *eof()* will become true once the queue is drained)

virtual void **copy_to**(*StreamPortBase* &other) = 0

virtual void **move_to**(*StreamPortBase* &other) = 0

virtual void **clear**() = 0

virtual *ProcessPtr* **get_buffer_writer**(const std::string &name) = 0

get a process with a compatible streaming input and a non-streaming output compatible with *get_buffer_writer()*, which writes the inputs to a buffer

virtual *ProcessPtr* **get_buffer_reader**(const std::string &name) = 0

get a process with a compatible streaming output and a non-streaming input compatible with *get_buffer_reader()*, which reads from a buffer

### Class ValidationError

- Defined in file_include_eat_framework_exceptions.hpp

### Inheritance Relationships

### Base Type

- `public runtime_error`

### Class Documentation

class **ValidationError** : public runtime_error

### Template Class ValuePtr

- Defined in file_include_eat_framework_value_ptr.hpp

### Class Documentation

template<typename **T**>

class **ValuePtr**

a wrapper around shared_ptr that has more value-like semantics while avoiding copies where possible

this should be used in ports (or structures moved through ports) to wrap things like ADM data which the user might want to modify in-place but are expensive to copy

the value can not be modified in-place, as this would be visible in other 'copies' of this structure

#### Public Functions

inline **ValuePtr**()

inline **ValuePtr**(std::shared_ptr<*T*> value_)

inline std::shared_ptr<const *T*> **read**() const

get read-only access to the value

inline std::shared_ptr<*T*> **move_or_copy**() const

get a non-const value that can be modified

this makes a copy if there are multiple users of the underlying value, or moves if there is only one

### Class BlockResampler

- Defined in file_include_eat_process_block_resampling.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::FunctionalAtomicProcess` (*Class FunctionalAtomicProcess*)

### Class Documentation

class **BlockResampler** : public eat::framework::*FunctionalAtomicProcess*

#### Public Functions

explicit **BlockResampler**(std::string const &name, adm::Time min_duration)

virtual void **process**() override

### Class BlockSubElementDropper

- Defined in file_include_eat_process_block_subelement_dropper.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::FunctionalAtomicProcess` (*Class FunctionalAtomicProcess*)

### Class Documentation

class **BlockSubElementDropper** : public eat::framework::*FunctionalAtomicProcess*

#### Public Types

enum **Droppable**

*Values:*

enumerator **Diffuse**

enumerator **ChannelLock**

enumerator **ObjectDivergence**

enumerator **JumpPosition**

enumerator **ScreenRef**

enumerator **Width**

enumerator **Depth**

enumerator **Height**

enumerator **Gain**

enumerator **Importance**

enumerator **Headlocked**

enumerator **HeadphoneVirtualise**

#### Public Functions

**BlockSubElementDropper**(std::string const &name, std::vector<*Droppable*> params_to_drop)

virtual void **process**() override

### Class InteractionLimiter

- Defined in file_include_eat_process_limit_interaction.hpp

### Nested Relationships

### Nested Types

- *Struct InteractionLimiter::Config*

### Inheritance Relationships

### Base Type

- `public eat::framework::FunctionalAtomicProcess` (*Class FunctionalAtomicProcess*)

### Class Documentation

class **InteractionLimiter** : public eat::framework::*FunctionalAtomicProcess*

#### Public Types

enum **Droppable**

*Values:*

enumerator **OnOff**

enumerator **Gain**

enumerator **Position**

#### Public Functions

explicit **InteractionLimiter**(std::string const &name, *Config* config)

virtual void **process**() override

struct **Config**

##### Public Members

bool **remove_disabled_ranges** = {false}

std::optional<*GainInteractionConstraint*> **gain_range**

std::optional<*PositionInteractionConstraint*> **position_range**

std::vector<*Droppable*> **types_to_disable**

### Class InterleavedSampleBlock

- Defined in file_include_eat_process_block.hpp

### Class Documentation

class **InterleavedSampleBlock**

a block of samples in which samples for each channel are interleaved

see also *PlanarSampleBlock*, which is equivalent but with planar (non-interleaved) channels

#### Public Functions

inline **InterleavedSampleBlock**(std::vector<float> samples, *BlockDescription* blockInfo)

construct with existing samples, which must have a size of sample_count * channel_count

inline **InterleavedSampleBlock**(*BlockDescription* blockInfo)

construct with zero-valued samples

inline *BlockDescription* const &**info**() const

get the block description (sample and channel count, sample rate)

inline float **sample**(size_t channel, size_t sample) const

access a single sample

inline float &**sample**(size_t channel, size_t sample)

access a single sample

inline const float ***data**() const

access the sample data

sample s of channel c is at *data()*[s * *info()*.channel_count + c]

inline float ***data**()

access the sample data

sample s of channel c is at *data()*[s * *info()*.channel_count + c]

### Class InterleavedStreamingAudioSink

- Defined in file_include_eat_process_block.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::StreamingAtomicProcess` (*Class StreamingAtomicProcess*)

### Class Documentation

class **InterleavedStreamingAudioSink** : public eat::framework::*StreamingAtomicProcess*

a sink for *InterleavedSampleBlock* which stores the samples, to be retrieved after a processing graph has completed

#### Public Functions

inline explicit **InterleavedStreamingAudioSink**(std::string const &name)

inline std::vector<float> const &**get**()

access the vector of samples

inline *InterleavedSampleBlock* **get_block**()

get the samples as an *InterleavedSampleBlock*

inline virtual void **initialise**() override

inline virtual void **process**() override

### Class InterleavedStreamingAudioSource

- Defined in file_include_eat_process_block.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::StreamingAtomicProcess` (*Class StreamingAtomicProcess*)

### Class Documentation

class **InterleavedStreamingAudioSource** : public eat::framework::*StreamingAtomicProcess*

a process which produces *InterleavedSampleBlock* objects from a buffer provided at initialisation

#### Public Functions

inline **InterleavedStreamingAudioSource**(std::string const &name, std::vector<float> samples, *BlockDescription* blockInfo)

construct with some samples

**Parameters**

- **samples** – interleaved samples
- **blockInfo** – shape of the produced blocks. channel_count and sample_rate will be kept as is, but sample_count is treated as the maximum number of samples to produce in one block (if the number of samples is not divisible by blockInfo.sample_count)

inline virtual void **process**() override

### Class JumpPositionRemover

- Defined in file_include_eat_process_jump_position_removal.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::FunctionalAtomicProcess` (*Class FunctionalAtomicProcess*)

### Class Documentation

class **JumpPositionRemover** : public eat::framework::*FunctionalAtomicProcess*

#### Public Functions

explicit **JumpPositionRemover**(std::string const &name)

virtual void **process**() override

### Class PlanarSampleBlock

- Defined in file_include_eat_process_block.hpp

### Class Documentation

class **PlanarSampleBlock**

a block of planar samples

see also *InterleavedSampleBlock*, which is equivalent but with interleaved channels

#### Public Functions

inline **PlanarSampleBlock**(std::vector<float> samples, *BlockDescription* blockInfo)

construct with existing samples, which must have a size of sample_count * channel_count

inline **PlanarSampleBlock**(*BlockDescription* blockInfo)

construct with zero-valued samples

inline *BlockDescription* const &**info**() const

get the block description (sample and channel count, sample rate)

inline float **sample**(size_t channel, size_t sample) const

access a single sample

inline float &**sample**(size_t channel, size_t sample)

access a single sample

inline const float *data() const

access the sample data

sample s of channel c is at *data()*[c * *info()*.sample_count + s]

inline float *data()

access the sample data

sample s of channel c is at *data()*[c * *info()*.sample_count + s]

### Class SilenceDetector

- Defined in file_include_eat_process_silence_detect.hpp

### Inheritance Relationships

### Base Type

- `public eat::framework::StreamingAtomicProcess` (*Class StreamingAtomicProcess*)

### Class Documentation

class **SilenceDetector** : public eat::framework::*StreamingAtomicProcess*

#### Public Functions

inline explicit **SilenceDetector**(std::string const &name, *SilenceDetectionConfig* config = {})

inline virtual void **initialise**() override

inline virtual void **process**() override

inline virtual void **finalise**() override

### Class SilenceStatus

- Defined in file_include_eat_process_silence_detect.hpp

### Class Documentation

class **SilenceStatus**

#### Public Functions

inline explicit **SilenceStatus**(*SilenceDetectionConfig* silence_config)

inline void **process**(*InterleavedSampleBlock* &block, std::size_t sample_number)

inline bool **ready**() const

inline *AudioInterval* **getInterval**() const

inline void **finish**()

### Class TempDir

- Defined in file_include_eat_process_temp_dir.hpp

### Class Documentation

class **TempDir**

a uniquely-named temporary directory in which temporary files can be created

currently this will be cleaned up at program exit, however it may be changed to clean up once all instances have gone out of scope, so keep a reference to this while you're using it

#### Public Functions

**TempDir**()

std::filesystem::path **get_temp_file**(const std::string &extension)

### Class ProfileValidator

- Defined in file_include_eat_process_validate.hpp

### Class Documentation

class **ProfileValidator**

holds a list of checks which can be ran on some ADM data to yield some results

#### Public Functions

inline **ProfileValidator**(std::vector<*Check*> checks_)

*ValidationResults* **run**(const *ADMData* &adm) const

run the checks on some ADM data

### Class ItemSelectionError

- Defined in file_include_eat_render_rendering_items.hpp

### Inheritance Relationships

### Base Type

- `public runtime_error`

### Class Documentation

class **ItemSelectionError** : public runtime_error

parent for errors raised during item selection

### Class TempDir

- Defined in file_include_eat_testing_files.hpp

### Class Documentation

class **TempDir**

a temporary directory for use while testing

currently `tmpdir / "file.wav"` when running a test called `test_name` returns test_tmp/test_name/file.wav, and these files are not deleted

#### Public Functions

inline **TempDir**()

inline std::filesystem::path **operator/**(const std::string &fname)

get a file in the directory

### Class Visitable

- Defined in file_include_eat_utilities_element_visitor.hpp

#### Class Documentation

class **Visitable**

interface for values that are visitable using the visit functions below

##### Public Functions

inline virtual **~Visitable**()

inline virtual bool **visit**(const std::string &desc, const std::function<void(*VisitablePtr*)>&)

visit the sub-elements described by desc returns true if desc is valid forthis type of value

virtual std::any **as_any**() = 0

get the held value as a std::any

template<typename **T**>
inline auto **as_t**()

get the hald value

inline virtual std::string **get_description**()

get a description for this element

- for elements with an ID, returns the ID
- for single elements (e.g. name), return the name of the element (possibly shortened for elements whose name contains the parent element name)
- for repeated elements, returns the type and something which identifies them (e.g. the value itself or a name)

### 3.2.3 Enums

#### Enum LanguageCodeType

- Defined in file_include_eat_process_language_codes.hpp

#### Enum Documentation

enum **eat::process::LanguageCodeType**

*Values:*

enumerator **UNKNOWN**

enumerator **REGULAR**

enumerator **RESERVED**

enumerator **UNCODED**

enumerator **MULTIPLE**

enumerator **UNDETERMINED**

enumerator **NO_CONTENT**

enumerator **SPECIAL**

enumerator **ANY**

enumerator **NONE**

### 3.2.4 Functions

#### Function eat::admx::add_with_different_denominators

- Defined in file_include_eat_process_adm_time_extras.hpp

#### Function Documentation

inline adm::FractionalTime `eat::admx::`**add_with_different_denominators**(adm::FractionalTime const &lhs, adm::FractionalTime const &rhs)

#### Function eat::admx::add_with_same_denominators

- Defined in file_include_eat_process_adm_time_extras.hpp

#### Function Documentation

inline adm::FractionalTime `eat::admx::`**add_with_same_denominators**(adm::FractionalTime const &lhs, adm::FractionalTime const &rhs)

#### Function eat::admx::minus(adm::FractionalTime const&, adm::FractionalTime const&)

- Defined in file_include_eat_process_adm_time_extras.hpp

### Function Documentation

inline adm::FractionalTime `eat::admx::minus`(adm::FractionalTime const &lhs, adm::FractionalTime const &rhs)

### Function eat::admx::minus(adm::Time const&, adm::Time const&)

- Defined in file_include_eat_process_adm_time_extras.hpp

### Function Documentation

inline adm::Time `eat::admx::minus`(adm::Time const &first, adm::Time const &second)

### Function eat::admx::negate

- Defined in file_include_eat_process_adm_time_extras.hpp

### Function Documentation

inline adm::FractionalTime `eat::admx::negate`(adm::FractionalTime const &time)

### Function eat::admx::plus(adm::FractionalTime const&, adm::FractionalTime const&)

- Defined in file_include_eat_process_adm_time_extras.hpp

### Function Documentation

inline adm::FractionalTime `eat::admx::plus`(adm::FractionalTime const &lhs, adm::FractionalTime const &rhs)

### Function eat::admx::plus(adm::Time const&, adm::Time const&)

- Defined in file_include_eat_process_adm_time_extras.hpp

### Function Documentation

inline adm::Time `eat::admx::plus`(adm::Time const &first, adm::Time const &second)

### Template Function eat::admx::roundToFractional

- Defined in file_include_eat_process_adm_time_extras.hpp

### Function Documentation

template<typename **Rep**, typename **Period**>
adm::FractionalTime `eat::admx::`**`roundToFractional`**(std::chrono::duration<*Rep*, *Period*> rhs, int64_t target_denominator)

### Function eat::framework::always_assert

- Defined in file_include_eat_framework_exceptions.hpp

### Function Documentation

inline void `eat::framework::`**`always_assert`**(bool condition, const std::string &message)

### Template Function eat::framework::copy_shared_ptr

- Defined in file_include_eat_framework_value_ptr.hpp

### Function Documentation

template<typename **T**>
std::shared_ptr<*T*> `eat::framework::`**`copy_shared_ptr`**(const std::shared_ptr<*T*> &value)

extension point for copying data stored in a shared_ptr, for types that are always stored in shared_ptrs and therefore don't have a copy constructor

### Specialized Template Function eat::framework::copy_shared_ptr< adm::Document >

- Defined in file_include_eat_process_adm_bw64.hpp

### Function Documentation

template<>
std::shared_ptr<adm::Document> `eat::framework::`**`copy_shared_ptr`**<adm::Document>(const std::shared_ptr<adm::Document> &value)

### Function eat::framework::evaluate

- Defined in file_include_eat_framework_evaluate.hpp

### Function Documentation

void `eat::framework::`**`evaluate`**(const *Graph* &g)
evaluate a graph; equivalent to plan(g).run()

### Function eat::framework::flatten

- Defined in file_include_eat_framework_evaluate.hpp

### Function Documentation

*Graph* `eat::framework::`**`flatten`**(const *Graph* &g)

### Function eat::framework::plan

- Defined in file_include_eat_framework_evaluate.hpp

### Function Documentation

*Plan* `eat::framework::`**`plan`**(const *Graph* &g)
plan the evaluation of graph

### Function eat::framework::run_with_progress

- Defined in file_include_eat_framework_evaluate.hpp

### Function Documentation

void `eat::framework::`**`run_with_progress`**(const *Plan* &p)
run a plan while printing progress updates to the terminal

### Function eat::framework::validate

- Defined in file_include_eat_framework_evaluate.hpp

### Function Documentation

void `eat::framework::validate`(const *Graph* &g)

### Function eat::process::clear_id

- Defined in file_include_eat_process_block_modification.hpp

### Function Documentation

void `eat::process::clear_id`(adm::AudioBlockFormatObjects &object)

### Function eat::process::de_duplicate_zero_length_blocks

- Defined in file_include_eat_process_block_resampling.hpp

### Function Documentation

std::vector<adm::AudioBlockFormatObjects> `eat::process::de_duplicate_zero_length_blocks`(adm::BlockFormatsRange<a
blocks)

### Function eat::process::format_language_code_types

- Defined in file_include_eat_process_language_codes.hpp

### Function Documentation

std::string `eat::process::format_language_code_types`(*LanguageCodeType* type)

### Function eat::process::load_chna

- Defined in file_include_eat_process_chna.hpp

### Function Documentation

void `eat::process::load_chna`(adm::Document &document, *channel_map_t* &channel_map, const
bw64::ChnaChunk &chna)

add information from a CHNA chunk into an ADM document and channel map

### Function eat::process::make_add_block_rtimes

- Defined in file_include_eat_process_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_add_block_rtimes`**(const std::string &name)

ensure that blocks with a specified duration have an rtime

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_apply_channel_mapping

- Defined in file_include_eat_process_channel_mapping.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_apply_channel_mapping`**(const std::string &name)

apply a ChannelMapping to some samples

this can be used to rearrange or remove channels

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- in_channel_mapping (DataPort<ChannelMapping>) : channel mapping to apply
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_block_resampler

- Defined in file_include_eat_process_block_resampling.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_block_resampler`**(const std::string &name, std::string const &min_duration)

### Function eat::process::make_block_subelement_dropper

- Defined in file_include_eat_process_block_subelement_dropper.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_block_subelement_dropper`**(std::string const &name, std::vector<*BlockSubElementDropper*::*Droppable*> to_drop)

### Function eat::process::make_chna

- Defined in file_include_eat_process_chna.hpp

### Function Documentation

bw64::ChnaChunk `eat::process::`**`make_chna`**(const adm::Document &document, const *channel_map_t* &channel_map)

make a CHNA chunk for an ADM document and channel map

### Function eat::process::make_convert_track_stream_to_channel

- Defined in file_include_eat_process_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_convert_track_stream_to_channel`**(const std::string &name)

replace audioTrackUid->audioTrackFormat->audioStreamFormat->audioChannelFormat references with audioTrackUid->audioChannelFormat references

this doesn't remove any unused elements, so use RemoveUnusedElements after this

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_fix_block_durations

- Defined in file_include_eat_process_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_fix_block_durations`**(const std::string &name)

fix audioBlockFormat durations to match up with the next rtimes

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_fix_ds_frequency

- Defined in file_include_eat_process_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_fix_ds_frequency`**(const std::string &name)

add frequency information to DirectSpeakers blocks with LFE in their name

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_fix_stream_pack_refs

- Defined in file_include_eat_process_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_fix_stream_pack_refs`**(const std::string &name)

remove audioPackFormatIDRef in audioStreamFormats that are of type PCM and have an audioChannelFormatIDRef

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_infer_object_interact

- Defined in file_include_eat_process_profile_conversion_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_infer_object_interact`**(const std::string &name)

ensure that all audioObjects have an interact parameter based on the presence or absence of the audioObjectInteraction element

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

## Function eat::process::make_jump_position_remover

- Defined in file_include_eat_process_jump_position_removal.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_jump_position_remover`**(const std::string &name)

## Function eat::process::make_measure_loudness

- Defined in file_include_eat_process_loudness.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_measure_loudness`**(const std::string &name, const ear::Layout &layout)

a process which measures the loudness of input samples

- in_samples (StreamPort<InterleavedBlockPtr>) : input samples
- out_loudness (DataPort<adm::LoudnessMetadata>) : measured loudness

## Function eat::process::make_read_adm

- Defined in file_include_eat_process_adm_bw64.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_read_adm`**(const std::string &name, const std::string &path)

read ADM data from a BW64 ADM file

ports:

- out_axml (DataPort<ADMData>) : output ADM data

**Parameters** **`path`** – path to the file to read

### Function eat::process::make_read_adm_bw64

- Defined in file_include_eat_process_adm_bw64.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_read_adm_bw64`**(const std::string &name, const std::string &path, size_t block_size)

read samples and ADM data from a BW64 file

ports:

- out_axml (DataPort<ADMData>) : output ADM data
- out_samples (StreamPort<InterleavedBlockPtr>) : output samples

**Parameters**

- **`path`** – path to the file to read
- **`block_size`** – maximum number of samples in each output block

### Function eat::process::make_read_bw64

- Defined in file_include_eat_process_adm_bw64.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_read_bw64`**(const std::string &name, const std::string &path, size_t block_size)

read samples from a BW64 file

ports:

- out_samples (StreamPort<InterleavedBlockPtr>) : output samples

**Parameters**

- **`path`** – path to the file to read
- **`block_size`** – maximum number of samples in each output block

### Function eat::process::make_remove_elements

- Defined in file_include_eat_process_remove_elements.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_remove_elements`**(const std::string &name, *ElementIds* ids)

a process which removes the given elements

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

## Function eat::process::make_remove_importance

- Defined in file_include_eat_process_profile_conversion_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_remove_importance`**(const std::string &name)

remove importance values from all audioObjects, audioPackFormats and audioBlockFormats

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

## Function eat::process::make_remove_object_times_common_unsafe

- Defined in file_include_eat_process_profile_conversion_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_remove_object_times_common_unsafe`**(const std::string &name)

remove start and duration from audioObjects which only reference common definitions audioChannelFormats

this could cause rendering changes if there are non-zero samples outside the range of the audioObject, but should be safe on EPS output

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_remove_object_times_data_safe

- Defined in file_include_eat_process_profile_conversion_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_remove_object_times_data_safe`**(const std::string &name)

remove time/duration from audioObjects where it is safe to do so (doesn't potentially change the rendering) and can be done by only changing the metadata (no audio changes, no converting common definitions audioChannelFormats to real audioChannelFormats)

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_remove_silent_atu

- Defined in file_include_eat_process_profile_conversion_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_remove_silent_atu`**(const std::string &name)

replace silent audioTrackUID references in audioObjects with a real track that references a silent channel

ports:

- in_samples (StreamPort<InterleavedBlockPtr>) : input samples
- out_samples (StreamPort<InterleavedBlockPtr>) : output samples
- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_remove_unused

- Defined in file_include_eat_process_remove_unused.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_remove_unused`**(const std::string &name)

a process which removes unreferenced elements from an ADM document, and re-packs the channels to remove unreferenced channels

ports:

- in_samples (StreamPort<InterleavedBlockPtr>) : input samples
- out_samples (StreamPort<InterleavedBlockPtr>) : output samples
- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_remove_unused_elements

- Defined in file_include_eat_process_remove_unused.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_remove_unused_elements`**(const std::string &name)

a process which removes unreferenced elements from an ADM document

in contrast with make_remove_unused, this doesn't do anything with the audio, so can be useful if previous changes will not have affected the use of channels

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_rewrite_content_objects_emission

- Defined in file_include_eat_process_profile_conversion_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_rewrite_content_objects_emission`**(const std::string &name, int max_objects_depth = 2)

rewrite the programme-content-object structure to make it compatible with emission profile rules

this may drop audioContents or audioObjects that are too nested (only ones that have audioObject references), so any information which applies to the audioObjects below will be lost

max_objects_depth is the maximum nesting depth of any object, which is defined for an object as:

- 0 for objects which do not contain object references
- the maximum object depth of any referenced objects, plus 1 for example, of this is 0, object nesting is removed

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_set_content_dialogue_default

- Defined in file_include_eat_process_profile_conversion_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_set_content_dialogue_default`**(const std::string &name)

set missing audioContent dialogue values to mixed

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

## Function eat::process::make_set_position_defaults

- Defined in file_include_eat_process_profile_conversion_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_set_position_defaults`**(const std::string &name)

set position defaults

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

## Function eat::process::make_set_profiles

- Defined in file_include_eat_process_profile_conversion_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_set_profiles`**(const std::string &name, const std::vector<*profiles*::Profile> &profiles)

set the list of profiles that this document should conform to

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

## Function eat::process::make_set_programme_loudness

- Defined in file_include_eat_process_loudness.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_set_programme_loudness`**(const std::string &name, const adm::AudioProgrammeId &programme_id)

a process which sets the loudness of an audioProgramme with the given ID

- in_axml (DataPort<ADMData>) : input ADM data
- in_loudness (DataPort<adm::LoudnessMetadata>) : measured loudness
- out_axml (DataPort<ADMData>) : output ADM data

## Function eat::process::make_set_version

- Defined in file_include_eat_process_misc.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_set_version`**(const std::string &name, const std::string &version)

set the audioFormatExtended version

ports:

- in_axml (DataPort<ADMData>) : input ADM data
- out_axml (DataPort<ADMData>) : output ADM data

## Function eat::process::make_update_all_programme_loudnesses

- Defined in file_include_eat_process_loudness.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_update_all_programme_loudnesses`**(const std::string &name)

a process which measures the loudness of all audioProgrammes (by rendering them to 4+5+0) and updates the axml to match

- in_axml (DataPort<ADMData>) : input ADM data
- in_samples (StreamPort<InterleavedBlockPtr>) : input samples for in_axml
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::process::make_validate

- Defined in file_include_eat_process_validate_process.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_validate`**(const std::string &name, const profiles::*Profile* &profile)

a process which takes an ADM document, checks it against a profile, and prints any errors then raises an exception if any issues are found

ports:

- in_axml (DataPort<ADMData>) : input ADM data

### Function eat::process::make_write_adm_bw64

- Defined in file_include_eat_process_adm_bw64.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_write_adm_bw64`**(const std::string &name, const std::string &path)

write samples and ADM data to a BW64 file

ports:

- in_samples (StreamPort<InterleavedBlockPtr>) : input samples

**Parameters** `path` – path to the file to read

### Function eat::process::make_write_bw64

- Defined in file_include_eat_process_adm_bw64.hpp

### Function Documentation

framework::*ProcessPtr* `eat::process::`**`make_write_bw64`**(const std::string &name, const std::string &path)

write samples to a BW64 file

ports:

- in_samples (StreamPort<InterleavedBlockPtr>) : input samples

**Parameters** `path` – path to the file to read

### Function eat::process::only_object_type

- Defined in file_include_eat_process_block_modification.hpp

### Function Documentation

std::vector<std::shared_ptr<adm::AudioChannelFormat>> eat::process::**only_object_type**(std::vector<std::shared_ptr<adm::Au
const &input)

### Function eat::process::parse_droppable

- Defined in file_include_eat_process_block_subelement_dropper.hpp

### Function Documentation

std::vector<*BlockSubElementDropper*::*Droppable*> eat::process::**parse_droppable**(std::vector<std::string>
const &to_drop)

### Function eat::process::parse_language_code

- Defined in file_include_eat_process_language_codes.hpp

### Function Documentation

*LanguageCodeType* eat::process::**parse_language_code**(const std::string &code)

### Function eat::process::referenced_channel_formats

- Defined in file_include_eat_process_block_modification.hpp

### Function Documentation

std::vector<std::shared_ptr<adm::AudioChannelFormat>> eat::process::**referenced_channel_formats**(adm::Document
&doc)

### Function eat::process::remove_jump_position

- Defined in file_include_eat_process_jump_position_removal.hpp

### Function Documentation

std::vector<adm::AudioBlockFormatObjects> eat::process::**remove_jump_position**(adm::BlockFormatsRange<adm::AudioBlo
input_blocks)

### Function eat::process::resample_to_minimum_preserving_zero

- Defined in file_include_eat_process_block_resampling.hpp

### Function Documentation

std::vector<adm::AudioBlockFormatObjects> eat::process::**resample_to_minimum_preserving_zero**(adm::BlockFormatsRar
blocks,
adm::Time
const
&in-
ter-
val)

### Function eat::process::split

- Defined in file_include_eat_process_block_modification.hpp

### Function Documentation

std::pair<adm::AudioBlockFormatObjects, adm::AudioBlockFormatObjects> eat::process::**split**(std::optional<adm::AudioBlock
const
&prior_block,
adm::AudioBlockFormatObjec
const
&block_to_split,
adm::Rtime
const
&split_point)

### Function eat::process::validate_config

- Defined in file_include_eat_config_file_validate_config.hpp

### Function Documentation

void `eat::process::validate_config`(nlohmann::json const &config, std::ostream &err)

### Function eat::process::validation::any_messages

- Defined in file_include_eat_process_validate.hpp

### Function Documentation

bool `eat::process::validation::any_messages`(const *ValidationResults* &results)

are there any error messages in results?

### Function eat::process::validation::format_check

- Defined in file_include_eat_process_validate.hpp

### Function Documentation

std::string `eat::process::validation::format_check`(const *Check* &check)

get an english representation for a single check

### Function eat::process::validation::format_message

- Defined in file_include_eat_process_validate.hpp

### Function Documentation

std::string `eat::process::validation::format_message`(const *Message* &message)

get an english representation for a single message

### Function eat::process::validation::format_results

- Defined in file_include_eat_process_validate.hpp

### Function Documentation

void `eat::process::validation::format_results`(std::ostream &s, const *ValidationResults* &results, bool show_checks_without_messages = false)

format results to a stream

either prints all checks and results, or only those with any messages, depending on show_checks_without_messages

### Function eat::process::validation::make_emission_profile_validator

- Defined in file_include_eat_process_validate.hpp

### Function Documentation

*ProfileValidator* `eat::process::validation::`**`make_emission_profile_validator`**(int level)

build a validator for a given emission profile level

### Function eat::process::validation::make_profile_validator

- Defined in file_include_eat_process_validate.hpp

### Function Documentation

*ProfileValidator* `eat::process::validation::`**`make_profile_validator`**(const profiles::*Profile*&)

build a validator for a known profile

### Function eat::render::make_render

- Defined in file_include_eat_render_render.hpp

### Function Documentation

framework::*ProcessPtr* `eat::render::`**`make_render`**(const std::string &name, const ear::Layout &layout, size_t block_size, const *SelectionOptionsId* &options = {})

render input audio and samples to channels ports:

- in_axml (DataPort<ADMData>) : input ADM data
- in_samples (StreamPort<InterleavedBlockPtr>) : input samples
- out_axml (DataPort<ADMData>) : output ADM data

### Function eat::render::select_items

- Defined in file_include_eat_render_rendering_items.hpp

### Function Documentation

*SelectionResult* `eat::render::`**`select_items`**(const std::shared_ptr<adm::Document> &doc, const *SelectionOptions* &options = {})

### Function eat::render::selection_options_from_ids

- Defined in file_include_eat_render_rendering_items_options_by_id.hpp

### Function Documentation

*SelectionOptions* eat::render::**selection_options_from_ids**(const std::shared_ptr<adm::Document> &doc,
const *SelectionOptionsId* &options)

### Function eat::testing::files_equal

- Defined in file_include_eat_testing_files.hpp

### Function Documentation

inline bool eat::testing::**files_equal**(const std::string &fname_a, const std::string &fname_b)

are the contents of two files equal?

### Template Function eat::utilities::count_references

- Defined in file_include_eat_utilities_for_each_reference.hpp

### Function Documentation

template<typename **To**, typename **From**>
size_t eat::utilities::**count_references**(const std::shared_ptr<*From*> &el)

get the number of references from el to To elements

the reference type must exist in libadm

### Function eat::utilities::element_visitor::dotted_path

- Defined in file_include_eat_utilities_element_visitor.hpp

### Function Documentation

std::string eat::utilities::element_visitor::**dotted_path**(const std::vector<std::string> &desc)

format a path by joining the element with periods

### Function eat::utilities::element_visitor::format_path

- Defined in file_include_eat_utilities_element_visitor.hpp

### Function Documentation

std::string `eat::utilities::element_visitor::`**`format_path`**(const std::vector<std::string> &path)

format a path by concatenating the elements in reverse with " in " e.g. {"APR_1001", "name"} -> "name in APR_1001"

### Function eat::utilities::element_visitor::path_to_strings

- Defined in file_include_eat_utilities_element_visitor.hpp

### Function Documentation

std::vector<std::string> `eat::utilities::element_visitor::`**`path_to_strings`**(const *Path* &path)

turn a path into a list of strings using get_description

if the path has more than one element, the leading document element is dropped as this is normally obvious from context

### Function eat::utilities::element_visitor::visit(const VisitablePtr&, const std::vector<std::string>&, const std::function<void(const Path&path)>&)

- Defined in file_include_eat_utilities_element_visitor.hpp

### Function Documentation

void `eat::utilities::element_visitor::`**`visit`**(const *VisitablePtr* &start, const std::vector<std::string> &desc, const std::function<void(const *Path* &path)> &cb)

visit sub-elements of start based on the path described by desc; calls cb once for each element

### Function eat::utilities::element_visitor::visit(std::shared_ptr<adm::Document>, const std::vector<std::string>&, const std::function<void(const Path&path)>&)

- Defined in file_include_eat_utilities_element_visitor.hpp

### Function Documentation

void `eat::utilities::element_visitor::`**`visit`**(std::shared_ptr<adm::Document> document, const std::vector<std::string> &desc, const std::function<void(const *Path* &path)> &cb)

visit sub-elements of document based on the path described by desc; calls cb once for each element

visit sub-elements of document based on the path described by desc; calls cb once for each element

WARNING: this uses const_cast to remove the const of adm::Document — therefore the elements referenced in the path must be treated as if they were const

### Template Function eat::utilities::for_each_reference

- Defined in file_include_eat_utilities_for_each_reference.hpp

### Function Documentation

template<typename **Element**, typename **F**>
void `eat::utilities::`**for_each_reference**(const std::shared_ptr<*Element*> &el, *F* f)

call f on each element referenced by el

### Template Function eat::utilities::for_each_reference_t

- Defined in file_include_eat_utilities_for_each_reference.hpp

### Function Documentation

template<typename **To**, typename **From**, typename **F**>
void `eat::utilities::`**for_each_reference_t**(const std::shared_ptr<*From*> &el, *F* f)

call f on each element of type To referenced by el if there is no reference of this type, do nothing

### Function eat::utilities::graph_to_dot

- Defined in file_include_eat_utilities_to_dot.hpp

### Function Documentation

void `eat::utilities::`**graph_to_dot**(std::ostream &s, const framework::*Graph* &g, bool recursive = true)

print g in graphviz format for debugging or documentation

this can be turned into a png by piping it through dot like this:

./example | dot -Tpng -o out.png

or writing it to a .gv file and using:

dot -Tpng -o out.png in.gv

### Function eat::utilities::parse_id_variant

- Defined in file_include_eat_utilities_parse_id_variant.hpp

#### Function Documentation

adm::ElementIdVariant `eat::utilities::`**`parse_id_variant`**(const std::string &id)

#### Template Function eat::utilities::unwrap_named

- Defined in file_include_eat_utilities_unwrap_named.hpp

#### Function Documentation

**Warning:** doxygenfunction: Unable to resolve function "eat::utilities::unwrap_named" with arguments "(T&&)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Error in declarator or parameters-and-qualifiers Invalid C++ declaration: Expected identifier in nested name, got keyword: auto [error at 25] template<typename T> auto unwrap_named (T &&value) -> decltype(detail::UnwrapNamedType< std::remove_cvref_t< T >> ————————————-^ If the function has a return type: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 39] template<typename T> auto unwrap_named (T &&value) -> decltype(detail::UnwrapNamedType< std::remove_cvref_t< T >> ——————————————————^ If declarator-id: Invalid C++ declaration: Expected ')' after 'decltype(<expr>'. [error at 113] template<typename T> auto unwrap_named (T &&value) -> decltype(detail::UnwrapNamedType< std::remove_cvref_t< T >> ————————————————————————————————————————————————————————^

#### Template Function eat::utilities::unwrap_shared

- Defined in file_include_eat_utilities_unwrap_shared.hpp

#### Function Documentation

**Warning:** doxygenfunction: Unable to resolve function "eat::utilities::unwrap_shared" with arguments "(T&&)". Candidate function could not be parsed. Parsing error is Error when parsing function declaration. If the function has no return type: Error in declarator or parameters-and-qualifiers Invalid C++ declaration: Expected identifier in nested name, got keyword: auto [error at 25] template<typename T> auto unwrap_shared (T &&value) -> decltype(detail::UnwrapShared< std::remove_cvref_t< T >> ————————————-^ If the function has a return type: Error in declarator or parameters-and-qualifiers If pointer to member declarator: Invalid C++ declaration: Expected '::' in pointer to member (function). [error at 40] template<typename T> auto unwrap_shared (T &&value) -> decltype(detail::UnwrapShared< std::remove_cvref_t< T >> ——————————————————-^ If declarator-id: Invalid C++ declaration: Expected ')' after 'decltype(<expr>'. [error at 111] template<typename T> auto unwrap_shared (T &&value) -> decltype(detail::UnwrapShared< std::remove_cvref_t< T >> ————————————————————————————————————————————————————————^

### 3.2.5 Variables

#### Variable eat::process::language_codes

- Defined in file_include_eat_process_language_codes.hpp

#### Variable Documentation

const std::set<std::string> `eat::process::language_codes`

### 3.2.6 Typedefs

#### Typedef eat::framework::AtomicProcessPtr

- Defined in file_include_eat_framework_process.hpp

#### Typedef Documentation

using `eat::framework::AtomicProcessPtr` = std::shared_ptr<*AtomicProcess*>

#### Typedef eat::framework::CompositeProcessPtr

- Defined in file_include_eat_framework_process.hpp

#### Typedef Documentation

using `eat::framework::CompositeProcessPtr` = std::shared_ptr<*CompositeProcess*>

#### Typedef eat::framework::DataPortBasePtr

- Defined in file_include_eat_framework_process.hpp

#### Typedef Documentation

using `eat::framework::DataPortBasePtr` = std::shared_ptr<*DataPortBase*>

#### Typedef eat::framework::DataPortPtr

- Defined in file_include_eat_framework_process.hpp

### Typedef Documentation

using `eat::framework::DataPortPtr` = std::shared_ptr<*DataPort*<T>>

### Typedef eat::framework::ExecStepPtr

- Defined in file_include_eat_framework_evaluate.hpp

### Typedef Documentation

using `eat::framework::ExecStepPtr` = std::shared_ptr<*ExecStep*>

### Typedef eat::framework::FunctionalAtomicProcessPtr

- Defined in file_include_eat_framework_process.hpp

### Typedef Documentation

using `eat::framework::FunctionalAtomicProcessPtr` = std::shared_ptr<*FunctionalAtomicProcess*>

### Typedef eat::framework::GraphPtr

- Defined in file_include_eat_framework_process.hpp

### Typedef Documentation

using `eat::framework::GraphPtr` = std::shared_ptr<*Graph*>

### Typedef eat::framework::PortPtr

- Defined in file_include_eat_framework_process.hpp

### Typedef Documentation

using `eat::framework::PortPtr` = std::shared_ptr<*Port*>

### Typedef eat::framework::ProcessPtr

- Defined in file_include_eat_framework_process.hpp

### Typedef Documentation

using `eat::framework::ProcessPtr` = std::shared_ptr<*Process*>

### Typedef eat::framework::StreamingAtomicProcessPtr

- Defined in file_include_eat_framework_process.hpp

### Typedef Documentation

using `eat::framework::StreamingAtomicProcessPtr` = std::shared_ptr<*StreamingAtomicProcess*>

### Typedef eat::framework::StreamPortBasePtr

- Defined in file_include_eat_framework_process.hpp

### Typedef Documentation

using `eat::framework::StreamPortBasePtr` = std::shared_ptr<*StreamPortBase*>

### Typedef eat::framework::StreamPortPtr

- Defined in file_include_eat_framework_process.hpp

### Typedef Documentation

using `eat::framework::StreamPortPtr` = std::shared_ptr<*StreamPort*<T>>

### Typedef eat::process::channel_map_t

- Defined in file_include_eat_process_chna.hpp

### Typedef Documentation

using `eat::process::channel_map_t` = std::map<adm::AudioTrackUidId, size_t>

mapping from audioTrackUids to zero-based channel numbers in the associated stream/file

### Typedef eat::process::ChannelMapping

- Defined in file_include_eat_process_channel_mapping.hpp

### Typedef Documentation

using `eat::process::ChannelMapping` = std::vector<size_t>

instruction for remapping audio channels

the size of this is the number of elements in the output, and `cm[output_channel] == input_channel`

### Typedef eat::process::ElementIds

- Defined in file_include_eat_process_remove_elements.hpp

### Typedef Documentation

using `eat::process::ElementIds` = std::vector<adm::ElementIdVariant>

### Typedef eat::process::InterleavedBlockPtr

- Defined in file_include_eat_process_block.hpp

### Typedef Documentation

using `eat::process::InterleavedBlockPtr` = framework::*ValuePtr*<*InterleavedSampleBlock*>

pointer to an interleaved sample block

### Typedef eat::process::PlanarBlockPtr

- Defined in file_include_eat_process_block.hpp

### Typedef Documentation

using `eat::process::`**`PlanarBlockPtr`** = framework::*ValuePtr*<*PlanarSampleBlock*>

pointer to a planar sample block

### Typedef eat::process::profiles::Profile

- Defined in file_include_eat_process_profiles.hpp

### Typedef Documentation

using `eat::process::profiles::`**`Profile`** = std::variant<*ITUEmissionProfile*>

represents some known profile, used to select behaviours for different profiles, for example when validating or conforming files

### Typedef eat::process::validation::Check

- Defined in file_include_eat_process_validate.hpp

### Typedef Documentation

using `eat::process::validation::`**`Check`** = std::variant<*ElementInList*<std::string>, *ElementInRange*<float>, *ElementPresent*, *NumElements*, *ObjectContentOrNested*, *StringLength*, *UniqueElements*<std::string>, *ValidLanguage*>

known checks

### Typedef eat::process::validation::CountRange

- Defined in file_include_eat_process_validate.hpp

### Typedef Documentation

using `eat::process::validation::`**`CountRange`** = *Range*<size_t>

### Typedef eat::process::validation::Message

- Defined in file_include_eat_process_validate.hpp

### Typedef Documentation

using `eat::process::validation::Message` = detail::ToMessages<*Check*>

messages that known checks can produce

### Typedef eat::process::validation::ValidationResults

- Defined in file_include_eat_process_validate.hpp

### Typedef Documentation

using `eat::process::validation::ValidationResults` = std::vector<*ValidationResult*>

results of all checks that have been ran

### Typedef eat::render::ContentIdStart

- Defined in file_include_eat_render_rendering_items_options_by_id.hpp

### Typedef Documentation

using `eat::render::ContentIdStart` = std::vector<adm::AudioContentId>

### Typedef eat::render::ContentStart

- Defined in file_include_eat_render_rendering_items.hpp

### Typedef Documentation

using `eat::render::ContentStart` = std::vector<std::shared_ptr<adm::AudioContent>>

### Typedef eat::render::ObjectIdStart

- Defined in file_include_eat_render_rendering_items_options_by_id.hpp

### Typedef Documentation

using `eat::render::ObjectIdStart` = std::vector<adm::AudioObjectId>

### Typedef eat::render::ObjectStart

- Defined in file_include_eat_render_rendering_items.hpp

### Typedef Documentation

using `eat::render::ObjectStart` = std::vector<std::shared_ptr<adm::AudioObject>>

### Typedef eat::render::ProgrammeIdStart

- Defined in file_include_eat_render_rendering_items_options_by_id.hpp

### Typedef Documentation

using `eat::render::ProgrammeIdStart` = adm::AudioProgrammeId

### Typedef eat::render::ProgrammeStart

- Defined in file_include_eat_render_rendering_items.hpp

### Typedef Documentation

using `eat::render::ProgrammeStart` = std::shared_ptr<adm::AudioProgramme>

### Typedef eat::render::SelectionStart

- Defined in file_include_eat_render_rendering_items.hpp

### Typedef Documentation

using `eat::render::SelectionStart` = std::variant<*DefaultStart*, *ProgrammeStart*, *ContentStart*, *ObjectStart*>

start point for item selection — either the default (first programme, all objects, or all tracks), a specific programme, a set of contents, or a set of objects

### Typedef eat::render::SelectionStartId

- Defined in file_include_eat_render_rendering_items_options_by_id.hpp

### Typedef Documentation

using `eat::render::`**`SelectionStartId`** = std::variant<*DefaultStart*, *ProgrammeIdStart*, *ContentIdStart*, *ObjectIdStart*>

### Typedef eat::render::TrackSpec

- Defined in file_include_eat_render_rendering_items.hpp

### Typedef Documentation

using `eat::render::`**`TrackSpec`** = std::variant<*DirectTrackSpec*, *SilentTrackSpec*>

### Typedef eat::utilities::element_visitor::Path

- Defined in file_include_eat_utilities_element_visitor.hpp

### Typedef Documentation

using `eat::utilities::element_visitor::`**`Path`** = std::vector<*VisitablePtr*>

### Typedef eat::utilities::element_visitor::VisitablePtr

- Defined in file_include_eat_utilities_element_visitor.hpp

### Typedef Documentation

using `eat::utilities::element_visitor::`**`VisitablePtr`** = std::shared_ptr<*Visitable*>

### Typedef eat::utilities::unwrap_named_t

- Defined in file_include_eat_utilities_unwrap_named.hpp

### Typedef Documentation

using `eat::utilities::`**`unwrap_named_t`** = typename detail::UnwrapNamedType<T>::type

get the inner type of a NamedType, or pass through T

CHAPTER

FOUR

# EXAMPLE CONFIGURATION FILES

- `fix_dolby.json`

  fix issues in Dolby Atmos ADM Profile files to be compatible with the BS.2127 renderer

  runs *fix_ds_frequency*, *fix_block_durations* and *fix_stream_pack_refs*.

  required options: `input.path` and `output.path`

- `track_stream_to_channel.json`

  replace audioTrackUid->audioTrackFormat->audioStreamFormat->audioChannelFormat references with audioTrackUid->audioChannelFormat references

  runs *convert_track_stream_to_channel* and *remove_unused_elements*

  required options: `input.path` and `output.path`

- `render.json`

  render an ADM BW64 file

  runs *add_block_rtimes* and *render*

  required options: `input.path`, `render.layout` and `output.path`

- `measure_loudness.json`

  update the loudness of one audioProgramme in a BW64 wav file

  this currently works by rendering the programme to 4+5+0 and measuring the loudness of the output, so this may not be accurate for channel-based content which would be better measured directly

  required options: `input.path` and `output.path`

- `measure_all_loudness.json`

  update the loudness of all audioProgrammes in a BW64 wav file

  this currently works by rendering the programme to 4+5+0 and measuring the loudness of the output, so this may not be accurate for channel-based content which would be better measured directly

  required options: `input.path` and `output.path`

- `conform_block_timing.json`

  conform the timing of object type audioblockformats to the emission profile

  this first removes the jump position flag from any audioblockformats in which it is present. these blocks are replaced with multiple blocks representing the same change where appropriate. it then ensures no block other than the first is shorter than 5ms in duration by combining short blocks, starting from the end.

  required options: `input.path` and `output.path`

- `conform_to_emission.json`

  read and write an ADM BW64 file, applying various processes to make it closer to the emission profile

  required options: `input.path` and `output.path`

- `validate_emission.json`

  read an ADM BW64 file and check it for compatibility with the emission profile

  required options: `input.path`

CHAPTER

FIVE

# INDICES AND TABLES

- genindex
- modindex
- search

# PROCESS INDEX

# INDEX

## F



## I



## M



## R



## S



## U



## V



## W