MilliSuono

Generated by Doxygen 1.15.0

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 ms::Event Struct Reference

Represents a time-stamped event in the audio processing timeline.

`#include <Port.hpp>`

Public Member Functions

- Event (const std::string &type, const ControlValue &value, int sampleOffset)

  Constructs an Event object.

Public Attributes

- std::string type
- ControlValue value
- int sampleOffset

3.1.1 Detailed Description

Represents a time-stamped event in the audio processing timeline.

Events are typically generated by control sources (e.g., user interaction, automation, or MIDI input) and scheduled at a specific sample offset within a processing block.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Event()

```
ms::Event::Event (
            const std::string & type,
            const ControlValue & value,
            int sampleOffset)  [inline]
```

Constructs an Event object.

Parameters

type	The event type identifier.
value	The payload associated with the event.
sampleOffset	The sample index relative to the start of the processing block.

3.1.3 Member Data Documentation

3.1.3.1 sampleOffset

```
int ms::Event::sampleOffset
```

The sample offset within the current processing block at which the event occurs.

3.1.3.2 type

```
std::string ms::Event::type
```

Type or category of the event (e.g., "note_on", "param_change").

3.1.3.3 value

```
ControlValue ms::Event::value
```

The event payload, which can be any supported ControlValue type.

The documentation for this struct was generated from the following file:

- include/core/Port.hpp

3.2 ms::Node Class Reference

Represents a processing unit in the MilliSuono graph.

```
#include <Node.hpp>
```

Public Member Functions

- Node (const std::string &id)
  Constructs a Node with a given identifier.
- virtual ~**Node** ()=default
  Virtual destructor for proper cleanup in derived classes.
- const std::string & getId () const
  Returns the unique identifier of the Node.
- const std::vector< Param > & getParams () const
  Returns the list of parameters associated with the Node (read-only).
- std::vector< Param > getParams ()
  Returns the list of parameters associated with the Node (mutable).
- const ControlValue * getParam (const std::string &name) const
  Retrieves a parameter value by name.
- void setParams (const std::vector< Param > &newParams)
  Sets the parameters of the Node.
- bool setParam (const std::string &name, const ControlValue &value)
  Sets a parameter value by name.

3.2.1 Detailed Description

Represents a processing unit in the MilliSuono graph.

A Node defines a functional unit (e.g. an oscillator, filter, or mixer) with a unique identifier and a set of configurable parameters. Nodes can be connected via Ports to form complex audio processing graphs.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Node()

```
ms::Node::Node (
            const std::string & id)  [inline]
```

Constructs a Node with a given identifier.

Parameters

id	The unique string identifier for the Node.

3.2.3 Member Function Documentation

3.2.3.1 getId()

```
const std::string & ms::Node::getId () const  [inline]
```

Returns the unique identifier of the Node.

Returns

The Node's identifier string.

3.2.3.2 getParam()

```
const ControlValue * ms::Node::getParam (
            const std::string & name) const  [inline]
```

Retrieves a parameter value by name.

Parameters

name	The name of the parameter to retrieve.

Returns

A pointer to the ControlValue if found, nullptr otherwise.

3.2.3.3 getParams() [1/2]

```
std::vector< Param > ms::Node::getParams ()  [inline]
```

Returns the list of parameters associated with the Node (mutable).

Returns

A const reference to the vector of Params.

3.2.3.4 getParams() [2/2]

```
const std::vector< Param > & ms::Node::getParams () const  [inline]
```

Returns the list of parameters associated with the Node (read-only).

Returns

A const reference to the vector of Params.

3.2.3.5 setParam()

```
bool ms::Node::setParam (
            const std::string & name,
            const ControlValue & value)  [inline]
```

Sets a parameter value by name.

Parameters

name	The name of the parameter to set.
value	The new value to assign to the parameter.

Returns

True if the parameter was found and set, false otherwise.

3.2.3.6 setParams()

```
void ms::Node::setParams (
            const std::vector< Param > & newParams)  [inline]
```

Sets the parameters of the Node.

Parameters

newParams	A vector of Params to set for the Node.

The documentation for this class was generated from the following file:

- include/core/Node.hpp

3.3 ms::Param Struct Reference

Represents a named parameter of a Node.

`#include <Node.hpp>`

Public Member Functions

- Param (const std::string &paramName, const ControlValue &paramValue)

Public Attributes

- std::string name
- ControlValue value

3.3.1 Detailed Description

Represents a named parameter of a Node.

A parameter stores a name and a corresponding ControlValue. Parameters can represent any configurable property such as gain, frequency, or mode.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Param()

```
ms::Param::Param (
            const std::string & paramName,
            const ControlValue & paramValue)  [inline]
```

Constuctor for convenience.

3.3.3 Member Data Documentation

3.3.3.1 name

```
std::string ms::Param::name
```

The unique name identifying the paraemter.

3.3.3.2 value

```
ControlValue ms::Param::value
```

The current value of the parameter.

The documentation for this struct was generated from the following file:

- include/core/Node.hpp

3.4 ms::Port Struct Reference

Represents an input or output port of a Node.

```
#include <Port.hpp>
```

Public Member Functions

- Port (const std::string &name, PortType type)
  Constructs a Port object.

Public Attributes

- std::string name
- PortType type

3.4.1 Detailed Description

Represents an input or output port of a Node.

Ports define the interface through which nodes exchange audio, control, or event data in the MilliSuono engine.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Port()

```
ms::Port::Port (
            const std::string & name,
            PortType type)  [inline]
```

Constructs a Port object.

Parameters

name	The name identifying the port.
type	The port type (Audio, Control, or Event).

3.4.3 Member Data Documentation

3.4.3.1 name

```
std::string ms::Port::name
```

The unique name of the port within a node.

3.4.3.2 type

```
PortType ms::Port::type
```

The type of the port (Audio, Control, or Event).

The documentation for this struct was generated from the following file:

- include/core/Port.hpp

Chapter 4

File Documentation

4.1 include/core/Node.hpp File Reference

Defines the Node and Parameter structures for the MilliSuono system.

```
#include "Port.hpp"
#include <string>
#include <unordered_map>
#include <vector>
```

Classes

- struct ms::Param

  Represents a named parameter of a Node.
- class ms::Node

  Represents a processing unit in the MilliSuono graph.

4.1.1 Detailed Description

Defines the Node and Parameter structures for the MilliSuono system.

It provides the core data strctures for representing processing units and their configurable parameters within the MilliSuono framework.

4.2 Node.hpp

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Port.hpp"
00003 #include <string>
00004 #include <unordered_map>
00005 #include <vector>
00006
00014
00015 namespace ms {
00016
00024 struct Param {
00026   std::string name;
00028   ControlValue value;
00029
00031   Param(const std::string &paramName, const ControlValue &paramValue)
00032       : name(paramName), value(paramValue) {}
00033 };
00034
00042 class Node {
00043 public:
00048   Node(const std::string &id) : id_(id) {}
00049
00053   virtual ~Node() = default;
00054
00059   const std::string &getId() const { return id_; }
00060
00065   const std::vector<Param> &getParams() const { return params_; }
00066
00071   std::vector<Param> getParams() { return params_; }
00072
00078   const ControlValue *getParam(const std::string &name) const {
00079     for (const auto &param : params_) {
00080       if (param.name == name) {
00081         return &param.value;
00082       }
00083     }
00084     return nullptr;
00085   }
00086
00091   void setParams(const std::vector<Param> &newParams) { params_ = newParams; }
00092
00099   bool setParam(const std::string &name, const ControlValue &value) {
00100     for (auto &param : params_) {
00101       if (param.name == name) {
00102         param.value = value;
00103         return true;
00104       }
00105     }
00106     return false;
00107   }
00108
00109 private:
00111   const std::string id_;
00113   std::vector<Param> params_;
00114 };
00115
00116 } // namespace ms
```

4.3 include/core/Port.hpp File Reference

Defines the basic data structures for ports and events in the MilliSuono system.

```
#include <string>
#include <variant>
```

Classes

- struct ms::Event

  Represents a time-stamped event in the audio processing timeline.
- struct ms::Port

  Represents an input or output port of a Node.

Typedefs

- using ms::ControlValue = std::variant<float, int, bool, std::string>

  Represents the value carried by a control or event port.

Enumerations

- enum class ms::PortType { **Audio** , **Control** , **Event** }

  Defines the possible types of ports in the MilliSuono system.

4.3.1 Detailed Description

Defines the basic data structures for ports and events in the MilliSuono system.

This file declares the fundamental types used for representing audio, control, and event connections within the MilliSuono framework.

4.3.2 Typedef Documentation

4.3.2.1 ControlValue

```
using ms::ControlValue = std::variant<float, int, bool, std::string>
```

Represents the value carried by a control or event port.

This can be one of the following:

- float: for continuous parameters (e.g., gain, frequency)
- int: for discrete parameters or indices
- bool: for binary control signals (e.g., mute, toggle)
- std::string: for symbolic or textual data

4.3.3 Enumeration Type Documentation

4.3.3.1 PortType

```
enum class ms::PortType  [strong]
```

Defines the possible types of ports in the MilliSuono system.

- Audio: for audio signal connections
- Control: for control parameters (float, int, bool, string)
- Event: for time-stamped control or trigger events

4.4 Port.hpp

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <string>
00003 #include <variant>
00004
00013
00014 namespace ms {
00015
00023 enum class PortType { Audio, Control, Event };
00024
00034 using ControlValue = std::variant<float, int, bool, std::string>;
00035
00043 struct Event {
00045   std::string type;
00046
00048   ControlValue value;
00049
00052   int sampleOffset;
00053
00061   Event(const std::string &type, const ControlValue &value, int sampleOffset)
00062       : type(type), value(value), sampleOffset(sampleOffset) {}
00063 };
00064
00071 struct Port {
00073   std::string name;
00074
00076   PortType type;
00077
00083   Port(const std::string &name, PortType type) : name(name), type(type) {}
00084 };
00085
00086 } // namespace ms
```

Index