Class OperatorSpec

Table of contents

- Defined in File operator_spec.hpp

Inheritance Relationships

Base Type

- `public holoscan::ComponentSpec` (Class ComponentSpec)

Class Documentation

class OperatorSpec : public holoscan::ComponentSpec

Class to define the specification of an operator.

Public Functions

inline explicit OperatorSpec(Fragment *fragment = nullptr)

Construct a new OperatorSpec object.

Parameters

fragment – The pointer to the fragment that contains this operator.

inline std::unordered_map<std::string, std::shared_ptr<IOSpec>> &inputs()

Get input specifications of this operator.

Returns

The reference to the input specifications of this operator.

template<typename DataT>
inline IOSpec &input()

Define an input specification for this operator.

Template Parameters

DataT – The type of the input data.

Returns

The reference to the input specification.

template<typename DataT>
inline IOSpec &input(std::string name)

Define an input specification for this operator.

Template Parameters

DataT – The type of the input data.

Parameters

name – The name of the input specification.

Returns

The reference to the input specification.

inline std::unordered_map<std::string, std::shared_ptr<IOSpec>> &outputs()

Get output specifications of this operator.

Returns

The reference to the output specifications of this operator.

template<typename DataT>
inline IOSpec &output()

Define an output specification for this operator.

Template Parameters

DataT – The type of the output data.

Returns

The reference to the output specification.

template<typename DataT>
inline IOSpec &output(std::string name)

Define an output specification for this operator.

Template Parameters

DataT – The type of the output data.

Parameters

name – The name of the output specification.

Returns

The reference to the output specification.

inline void param(Parameter<holoscan::IOSpec*> &parameter, const char *key, const char *headline, const char *description, ParameterFlag flag = ParameterFlag::kNone)

Define an IOSpec* parameter for this operator.

Parameters

- **parameter** – The parameter to define.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **description** – The description of the parameter.
- **flag** – The flag of the parameter (default: ParameterFlag::kNone).

inline void param(Parameter<holoscan::IOSpec*> &parameter, const char *key, const char *headline, const char *description, std::initializer_list<void*> init_list)

Define an IOSpec* parameter for this operator.

This method is to catch the following case:

```
... spec.param(iospec1_, "iospec1", "IO Spec", "Example IO Spec.", {}); ...
private: Parameter<holoscan::IOSpec*> iospec1_;
```

Otherwise, `{}` will be treated as `ParameterFlag::kNone` instead of `std::initializer_list`.

Parameters

- **parameter** – The parameter to define.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **description** – The description of the parameter.
- **init_list** – The initializer list of the parameter.

inline void param(Parameter<holoscan::IOSpec*> &parameter, const char *key, const char *headline, const char *description, holoscan::IOSpec *default_value, ParameterFlag flag = ParameterFlag::kNone)

Define an IOSpec* parameter with a default value for this operator.

Parameters

- **parameter** – The parameter to define.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **description** – The description of the parameter.
- **default_value** – The default value of the parameter.
- **flag** – The flag of the parameter (default: ParameterFlag::kNone).

inline void param(Parameter<std::vector<holoscan::IOSpec*>> &parameter, const char *key, const char *headline, const char *description, ParameterFlag flag = ParameterFlag::kNone)

Define a IOSpec* vector parameter for this operator.

Parameters

- **parameter** – The parameter to define.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **description** – The description of the parameter.
- **flag** – The flag of the parameter (default: ParameterFlag::kNone).

inline void param(Parameter<std::vector<holoscan::IOSpec*>> &parameter, const char *key, const char *headline, const char *description, std::initializer_list<holoscan::IOSpec*> init_list)

Define a IOSpec* vector parameter for this operator.

This method is to catch the following case:

```
... spec.param(iospec1_, "iospec1", "IO Spec", "Example IO Spec.", {}); ...
private: Parameter<std::vector<holoscan::IOSpec*>> iospec1_;
```

Otherwise, `{}` will be treated as `ParameterFlag::kNone` instead of `std::initializer_list`.

Parameters

- **parameter** – The parameter to define.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **description** – The description of the parameter.
- **init_list** – The initializer list of the parameter.

inline void param(Parameter<std::vector<holoscan::IOSpec*>> &parameter, const char *key, const char *headline, const char *description, std::vector<holoscan::IOSpec*> default_value, ParameterFlag flag = ParameterFlag::kNone)

Define an IOSpec* parameter with a default value for this operator.

Parameters

- **parameter** – The parameter to define.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **description** – The description of the parameter.
- **default_value** – The default value of the parameter.
- **flag** – The flag of the parameter (default: ParameterFlag::kNone).

virtual YAML::Node to_yaml_node() const override

Get a YAML representation of the operator spec.

Returns

YAML node including the inputs, outputs, and parameters of this operator.

template<typename typeT>
inline void param(Parameter<typeT> &parameter, const char *key, ParameterFlag flag = ParameterFlag::kNone)

Define a parameter for this component.

Template Parameters

typeT – The type of the parameter.

Parameters

- **parameter** – The parameter to define.

- **key** – The key (name) of the parameter.
- **flag** – The flag of the parameter (default: ParameterFlag::kNone).

template<typename typeT>
inline void param(Parameter<typeT> &parameter, const char *key, const char *headline, ParameterFlag flag = ParameterFlag::kNone)

Define a parameter for this component.

Template Parameters

typeT – The type of the parameter.

Parameters

- **parameter** – The parameter to define.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **flag** – The flag of the parameter (default: ParameterFlag::kNone).

template<typename typeT>
void param(Parameter<typeT> &parameter, const char *key, const char *headline, const char *description, ParameterFlag flag = ParameterFlag::kNone)

Define a parameter for this component.

Template Parameters

typeT – The type of the parameter.

Parameters

- **parameter** – The parameter to define.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **description** – The description of the parameter.

- **flag** – The flag of the parameter (default: ParameterFlag::kNone).

template<typename typeT>
void param(Parameter<typeT> &parameter, const char *key, const char *headline, const char *description, std::initializer_list<void*> init_list)

Define a parameter for this component.

This method is to catch the following case:

> ... spec.param(int64_value_, "int64_param", "int64_t param", "Example int64_t parameter.", {}); ... private: Parameter<int64_t> int64_param_;

Otherwise, `{}` will be treated as `ParameterFlag::kNone` instead of `std::initializer_list`.

Template Parameters

typeT – The type of the parameter.

Parameters

- **parameter** – The parameter to define.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **description** – The description of the parameter.
- **init_list** – The initializer list of the parameter.

template<typename typeT>
void param(Parameter<typeT> &parameter, const char *key, const char *headline, const char *description, const typeT &default_value, ParameterFlag flag = ParameterFlag::kNone)

Define a parameter that has a default value.

Template Parameters

typeT – The type of the parameter.

Parameters

- **parameter** – The parameter to get.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **description** – The description of the parameter.
- **default_value** – The default value of the parameter.
- **flag** – The flag of the parameter (default: ParameterFlag::kNone).

template<typename typeT>
void param(Parameter<typeT> &parameter, const char *key, const char *headline, const char *description, typeT &&default_value, ParameterFlag flag = ParameterFlag::kNone)

Define a parameter that has a default value.

Template Parameters

typeT – The type of the parameter.

Parameters

- **parameter** – The parameter to get.
- **key** – The key (name) of the parameter.
- **headline** – The headline of the parameter.
- **description** – The description of the parameter.
- **default_value** – The default value of the parameter.
- **flag** – The flag of the parameter (default: ParameterFlag::kNone).

Protected Attributes

std::unordered_map<std::string, std::shared_ptr<IOSpec>> inputs_

Input specs.

std::unordered_map<std::string, std::shared_ptr<IOSpec>> outputs_

Outputs specs.