Class ComponentSpec

Table of contents

- Defined in File component_spec.hpp

Inheritance Relationships

Derived Type

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

Class Documentation

class ComponentSpec

Class to define the specification of a component.

Subclassed by holoscan::OperatorSpec

Public Functions

inline explicit ComponentSpec(Fragment *fragment = nullptr)

Construct a new ComponentSpec object.

Parameters

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

inline void fragment(Fragment *fragment)

Set the pointer to the fragment that contains this component.

Parameters

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

inline Fragment *fragment()

Get the pointer to the fragment that contains this component.

Returns

The pointer to the fragment that contains this component.

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).

inline std::unordered_map<std::string, ParameterWrapper> &params()

Get the parameters of this component.

Returns

The reference to the parameters of this component.

virtual YAML::Node to_yaml_node() const

Get a YAML representation of the component spec.

Returns

YAML node including the parameters of this component.

std::string description() const

Get a description of the component spec.

to_yaml_node()

Returns

YAML string.

Protected Attributes

Fragment *fragment_ = nullptr

The pointer to the fragment that contains this component.

std::unordered_map<std::string, ParameterWrapper> params_

The parameters of this component.