holoscan.core

This module provides a Python API for the core C++ API classes.

The *Application* class is the primary class that should be derived from to create a custom application.

`holoscan.core.Application`([argv])	Application class.
`holoscan.core.Arg`	Class representing a typed argument.
`holoscan.core.ArgContainerType`	Enum class for an *Arg*'s container type.
`holoscan.core.ArgElementType`	Enum class for an *Arg*'s element type.
`holoscan.core.ArgList`	Class representing a list of arguments.
`holoscan.core.ArgType`	Class containing argument type info.

holoscan.core.CLIOptions	Attributes
holoscan.core.Component	Base component class.
holoscan.core.ComponentSpec	alias of holoscan.core._core.PyComponentSpec
holoscan.core.ConditionType	Enum class for Condition types.
holoscan.core.Condition	Class representing a condition.
holoscan.core.Config	Configuration class.

holoscan.core.DataFlowMetric	Enum class for DataFlowMetric type.
holoscan.core.DataFlowTracker	Data Flow Tracker class.
holoscan.core.DLDevice	DLDevice class.
holoscan.core.DLDeviceType	Members:
holoscan.core.ExecutionContext	Class representing an execution context.
holoscan.core.Executor	Executor class.

holoscan.core.Fragment([app, name])	Fragment class.
holoscan.core.Graph	alias of holoscan.graphs._graphs.OperatorGraph
holoscan.core.InputContext	Class representing an input context.
holoscan.core.IOSpec	I/O specification class.
holoscan.core.Message	Class representing a message.
holoscan.core.NetworkContext	Class representing a network context.

`holoscan.core.Operator`(fragment, *args, **kwargs)	Operator class.
`holoscan.core.OperatorSpec`	alias of `holoscan.core._core.PyOperatorSpec`
`holoscan.core.OutputContext`	Class representing an output context.
`holoscan.core.ParameterFlag`	Enum class for parameter flags.
`holoscan.core.Resource`	Class representing a resource.
`holoscan.core.Tensor`	alias of `holoscan.core._core.PyTensor`

`holoscan.core.Tracker`(app, *[, filename, ...])	Context manager to add data flow tracking to an application.
`holoscan.core.arg_to_py_object`(arg)	Utility that converts an *Arg* to a corresponding Python object.
`holoscan.core.arglist_to_kwargs`(arglist)	Utility that converts an *ArgList* to a Python kwargs dictionary.
`holoscan.core.kwargs_to_arglist`(**kwargs)	Utility that converts a set of python keyword arguments to an *ArgList*.
`holoscan.core.py_object_to_arg`(obj[, name])	Utility that converts a single python argument to a corresponding *Arg* type.

class holoscan.core.Application(*argv=None*, **args*, ***kwargs*)

Bases: `holoscan.core._core.Application`

Application class.

This constructor parses the command line for flags that are recognized by App Driver/Worker, and removes all recognized flags so users can use the remaining flags for their own purposes.

If the arguments are not specified, the arguments are retrieved from `sys.executable` and `sys.argv`.

The arguments after processing arguments (parsing Holoscan-specific flags and removing them) are accessible through the `argv` attribute.

Parameters

argv

The command line arguments to parse. The first item should be the path to the python executable. If not specified, `[sys.executable, *sys.argv]` is used.

Examples

```
>>> from holoscan.core import Application >>> import sys >>>
Application().argv == sys.argv True >>> Application([]).argv == sys.argv True >>>
Application([sys.executable, *sys.argv]).argv == sys.argv True >>>
Application(["python3", "myapp.py", "--driver", "--address=10.0.0.1",
"my_arg1"]).argv ['myapp.py', 'my_arg1']
```

Attributes

`application`	The application associated with the fragment.
`argv`	The command line arguments after processing flags.

description	The application's description.
executor	Get the executor associated with the fragment.
fragment_graph	Get the computation graph (Graph node is a Fragment) associated with the application.
graph	Get the computation graph (Graph node is an Operator) associated with the fragment.
name	The fragment's name.
options	The reference to the CLI options.
version	The application's version.

Methods

add_flow (*args, **kwargs)	Overloaded function.
add_fragment (self, frag)	Add a fragment to the application.
add_operator (self, o	Add an operator to the application.

p)	
`compose` (self)	The compose method of the application.
`config` (*args, **kwargs)	Overloaded function.
`config_keys` (self)	The set of keys present in the fragment's configuration file.
`from_config` (self, key)	Retrieve parameters from the associated configuration.
`kwargs` (self, key)	Retrieve a dictionary parameters from the associated configuration.
`network_context` (*args, **kwargs)	Overloaded function.
`run` (self)	The run method of the application.
`run_async` ()	Run the application asynchronously.

`scheduler`(*args, **kwargs)	Overloaded function.
`track`(self, num_start_messages_to_skip, ...)	The track method of the application.

__init__(*self: holoscan.core._core.Application*, *argv: List[str] = []*) None

Application class.

This constructor parses the command line for flags that are recognized by App Driver/Worker, and removes all recognized flags so users can use the remaining flags for their own purposes.

If the arguments are not specified, the arguments are retrieved from `sys.executable` and `sys.argv`.

The arguments after processing arguments (parsing Holoscan-specific flags and removing them) are accessible through the `argv` attribute.

Parameters

argv

The command line arguments to parse. The first item should be the path to the python executable. If not specified, `[sys.executable, *sys.argv]` is used.

Examples

```
>>> from holoscan.core import Application >>> import sys >>> Application().argv == sys.argv True >>> Application([]).argv == sys.argv
```

```
True >>> Application([sys.executable, *sys.argv]).argv == sys.argv True
>>> Application(["python3", "myapp.py", "--driver", "--address=10.0.0.1",
"my_arg1"]).argv ['myapp.py', 'my_arg1']
```

add_flow(**args*, ***kwargs*)

Overloaded function.

1. add_flow(self: holoscan.core._core.Application, upstream_op: holoscan.core._core.Operator, downstream_op: holoscan.core._core.Operator) -> None

2. add_flow(self: holoscan.core._core.Application, upstream_op: holoscan.core._core.Operator, downstream_op: holoscan.core._core.Operator, port_pairs: Set[Tuple[str, str]]) -> None

Connect two operators associated with the fragment.

Parameters

upstream_op

Source operator.

downstream_op

Destination operator.

port_pairs

Sequence of ports to connect. The first element of each 2-tuple is a port from *upstream_op* while the second element is the port of *downstream_op* to which it connects.

Notes

This is an overloaded function. Additional variants exist:

1.) For the Application class there is a variant where the first two arguments are of type *holoscan.core.Fragment* instead of *holoscan.core.Operator*. This variant is used in building multi-fragment applications. 2.) There are also variants that omit the *port_pairs* argument that are applicable when there is

only a single output on the upstream operator/fragment and a single input on the downstream operator/fragment.

3. add_flow(self: holoscan.core._core.Application, upstream_frag: holoscan.core._core.Fragment, downstream_frag: holoscan.core._core.Fragment, port_pairs: Set[Tuple[str, str]]) -> None

add_fragment(*self: holoscan.core._core.Application*, *frag: holoscan.core._core.Fragment*) None

Add a fragment to the application.

Parameters

frag

The fragment to add.

add_operator(*self: holoscan.core._core.Application*, *op: holoscan.core._core.Operator*) None

Add an operator to the application.

Parameters

op

The operator to add.

property application

The application associated with the fragment.

Returns

app

property argv

The command line arguments after processing flags. This does not include the python executable like *sys.argv* does.

Returns

argv

compose(*self: holoscan.core._core.Application*) None

The compose method of the application.

This method should be called after *config*, but before *run* in order to compose the computation graph.

config(**args*, ***kwargs*)

Overloaded function.

1. config(self: holoscan.core._core.Fragment, config_file: str, prefix: str = '') -> None

Configuration class.

Represents configuration parameters as read from a YAML file.

Parameters

config

The path to the configuration file (in YAML format) or a *holoscan.core.Config* object.

prefix

Prefix path for the` config` file. Only available in the overloaded variant that takes a string for *config*.

2. config(self: holoscan.core._core.Fragment, arg0: holoscan.core._core.Config) -> None
3. config(self: holoscan.core._core.Fragment) -> holoscan.core._core.Config

config_keys(*self: holoscan.core._core.Fragment*) Set[str]

The set of keys present in the fragment's configuration file.

property description

The application's description.

Returns

description

property executor

Get the executor associated with the fragment.

property fragment_graph

Get the computation graph (Graph node is a Fragment) associated with the application.

from_config(*self: holoscan.core._core.Fragment*, *key: str*) object

Retrieve parameters from the associated configuration.

Parameters

key

The key within the configuration file to retrieve. This can also be a specific component of the parameter via syntax *'key.sub_key'*.

Returns

args

An argument list associated with the key.

property graph

Get the computation graph (Graph node is an Operator) associated with the fragment.

kwargs(*self: holoscan.core._core.Fragment*, *key: str*) dict

Retrieve a dictionary parameters from the associated configuration.

Parameters

key

The key within the configuration file to retrieve. This can also be a specific component of the parameter via syntax *'key.sub_key'*.

Returns

kwargs

A Python dict containing the parameters in the configuration file under the specified key.

property name

The fragment's name.

Returns

name

network_context(**args*, ***kwargs*)

Overloaded function.

1. network_context(self: holoscan.core._core.Fragment, network_context: holoscan.core._core.NetworkContext) -> None

Assign a network context to the Fragment

Parameters

network_context

A network_context class instance to be used by the underlying GXF executor. If unspecified, no network context will be used.

2. network_context(self: holoscan.core._core.Fragment) -> holoscan.core._core.NetworkContext
Get the network context to be used by the Fragment

property options

The reference to the CLI options.

Returns

options

run(*self: holoscan.core._core.Application*) None

The run method of the application.

This method runs the computation. It must have first been initialized via *config* and *compose*.

run_async()

Run the application asynchronously.

This method is a convenience method that creates a thread pool with one thread and runs the application in that thread. The thread pool is created using *concurrent.futures.ThreadPoolExecutor*.

Returns

future : concurrent.futures.Future

scheduler(**args*, ***kwargs*)

Overloaded function.

1. scheduler(self: holoscan.core._core.Fragment, scheduler: holoscan.core._core.Scheduler) -> None

Assign a scheduler to the Fragment.

Parameters

scheduler

A scheduler class instance to be used by the underlying GXF executor. If unspecified, the default is a *holoscan.gxf.GreedyScheduler*.

2. scheduler(self: holoscan.core._core.Fragment) -> holoscan.core._core.Scheduler
Get the scheduler to be used by the Fragment.

track(*self: holoscan.core._core.Fragment*, *num_start_messages_to_skip: int = 10*, *num_last_messages_to_discard: int = 10*, *latency_threshold: int = 0*)
holoscan::DataFlowTracker

The track method of the application.

This method enables data frame flow tracking and returns a DataFlowTracker object which can be used to display metrics data for profiling an application.

Parameters

num_start_messages_to_skip

The number of messages to skip at the beginning.

num_last_messages_to_discard

The number of messages to discard at the end.

latency_threshold

The minimum end-to-end latency in milliseconds to account for in the end-to-end latency metric calculations

property version

The application's version.

Returns

version

class holoscan.core.Arg

Bases: pybind11_builtins.pybind11_object

Class representing a typed argument.

Attributes

arg_type	ArgType info corresponding to the argument.

description	YAML formatted string describing the argument.
has_value	Boolean flag indicating whether a value has been assigned to the argument.
name	The name of the argument.

__init__(*self: holoscan.core._core.Arg*, *name: str*) None

Class representing a typed argument.

Parameters

name

The argument's name.

property arg_type

ArgType info corresponding to the argument.

Returns

arg_type

property description

YAML formatted string describing the argument.

property has_value

Boolean flag indicating whether a value has been assigned to the argument.

property name

The name of the argument.

Returns

name

class holoscan.core.ArgContainerType

Bases: pybind11_builtins.pybind11_object

Enum class for an *Arg*'s container type.

Members:

NATIVE

VECTOR

ARRAY

Attributes

name	

value	

ARRAY = *<ArgContainerType.ARRAY: 2>*

NATIVE = *<ArgContainerType.NATIVE: 0>*

VECTOR = *<ArgContainerType.VECTOR: 1>*

__init__(*self: holoscan.core._core.ArgContainerType*, *value: int*) None

property name

property value

class holoscan.core.ArgElementType

Bases: pybind11_builtins.pybind11_object

Enum class for an *Arg*'s element type.

Members:

CUSTOM

BOOLEAN

INT8

UNSIGNED8

INT16

UNSIGNED16

INT32

UNSIGNED32

INT64

UNSIGNED64

FLOAT32

FLOAT64

STRING

HANDLE

YAML_NODE

IO_SPEC

CONDITION

RESOURCE

Attributes

name	

value	

BOOLEAN = *<ArgElementType.BOOLEAN: 1>*

CONDITION = *<ArgElementType.CONDITION: 18>*

CUSTOM = *<ArgElementType.CUSTOM: 0>*

FLOAT32 = *<ArgElementType.FLOAT32: 10>*

FLOAT64 = *<ArgElementType.FLOAT64: 11>*

HANDLE = *<ArgElementType.HANDLE: 15>*

INT16 = *<ArgElementType.INT16: 4>*

INT32 = *<ArgElementType.INT32: 6>*

INT64 = *<ArgElementType.INT64: 8>*

INT8 = *<ArgElementType.INT8: 2>*

IO_SPEC = *<ArgElementType.IO_SPEC: 17>*

RESOURCE = *<ArgElementType.RESOURCE: 19>*

STRING = *<ArgElementType.STRING: 14>*

UNSIGNED16 = *<ArgElementType.UNSIGNED16: 5>*

UNSIGNED32 = *<ArgElementType.UNSIGNED32: 7>*

UNSIGNED64 = *<ArgElementType.UNSIGNED64: 9>*

UNSIGNED8 = *<ArgElementType.UNSIGNED8: 3>*

YAML_NODE = *<ArgElementType.YAML_NODE: 16>*

__init__(*self: holoscan.core._core.ArgElementType*, *value: int*) None

property name

property value

class holoscan.core.ArgList

Bases: pybind11_builtins.pybind11_object

Class representing a list of arguments.

Attributes

args	The underlying list of *Arg* objects.
description	YAML formatted string describing the list.
name	The name of the argument list.
size	The number of arguments in the list.

Methods

add(*args, **kwargs)	Overloaded function.
clear(self)	Clear the argument list.

__init__(*self: holoscan.core._core.ArgList*) None

Class representing a list of arguments.

add(**args*, ***kwargs*)

Overloaded function.

1. add(self: holoscan.core._core.ArgList, arg: holoscan.core._core.Arg) -> None

Add an argument to the list.

2. add(self: holoscan.core._core.ArgList, arg: holoscan.core._core.ArgList) -> None

Add a list of arguments to the list.

property args

The underlying list of *Arg* objects.

clear(*self: holoscan.core._core.ArgList*) None

Clear the argument list.

property description

YAML formatted string describing the list.

property name

The name of the argument list.

Returns

name

property size

The number of arguments in the list.

class holoscan.core.ArgType

Bases: pybind11_builtins.pybind11_object

Class containing argument type info.

Attributes

container_type	The container type of the argument.
dimension	The dimension of the argument container.
element_type	The element type of the argument.

to_string	String describing the argument type.

__init__(**args*, ***kwargs*)

Overloaded function.

1. __init__(self: holoscan.core._core.ArgType) -> None

Class containing argument type info.

2. __init__(self: holoscan.core._core.ArgType, element_type: holoscan.core._core.ArgElementType, container_type: holoscan.core._core.ArgContainerType) -> None

Class containing argument type info.

Parameters

element_type

Element type of the argument.

container_type

Container type of the argument.

property container_type

The container type of the argument.

property dimension

The dimension of the argument container.

property element_type

The element type of the argument.

property to_string

String describing the argument type.

class holoscan.core.CLIOptions

Bases: pybind11_builtins.pybind11_object

Attributes

config_path	The path to the configuration file.
driver_address	The address of the App Driver.
run_driver	The flag to run the App Driver.
run_worker	The flag to run the App Worker.
worker_address	The address of the App Worker.
worker_targets	The list of fragments for the App Worker.

Methods

print(self)	Print the CLI Options.

__init__(*self: holoscan.core._core.CLIOptions*, *run_driver: bool = False*, *run_worker: bool = False*, *driver_address: str = ''*, *worker_address: str = ''*, *worker_targets: List[str] = []*, *config_path: str = ''*) None

CLIOptions class.

property config_path

The path to the configuration file.

property driver_address

The address of the App Driver.

print(*self: holoscan.core._core.CLIOptions*) None

Print the CLI Options.

property run_driver

The flag to run the App Driver.

property run_worker

The flag to run the App Worker.

property worker_address

The address of the App Worker.

property worker_targets

The list of fragments for the App Worker.

Returns

worker_targets

class holoscan.core.Component

Bases: holoscan.core._core.ComponentBase

Base component class.

Attributes

args	The list of arguments associated with the component.

description	YAML formatted string describing the component.
fragment	The fragment containing the component.
id	The identifier of the component.
name	The name of the component.

Methods

add_arg(*args, **kwargs)	Overloaded function.
initialize(self)	Initialize the component.

__init__(*self: holoscan.core._core.Component*) None

Base component class.

add_arg(**args*, ***kwargs*)

Overloaded function.

1. add_arg(self: holoscan.core._core.ComponentBase, arg: holoscan.core._core.Arg) -> None

Add an argument to the component.

2. add_arg(self: holoscan.core._core.ComponentBase, arg: holoscan.core._core.ArgList) -> None

Add a list of arguments to the component.

property args

The list of arguments associated with the component.

Returns

arglist

property description

YAML formatted string describing the component.

property fragment

The fragment containing the component.

Returns

name

property id

The identifier of the component.

The identifier is initially set to `-1`, and will become a valid value when the component is initialized.

With the default executor (*holoscan.gxf.GXFExecutor*), the identifier is set to the GXF component ID.

Returns

id

initialize(*self: holoscan.core._core.ComponentBase*) None

Initialize the component.

property name

The name of the component.

Returns

name

holoscan.core.ComponentSpec

alias of `holoscan.core._core.PyComponentSpec`

class holoscan.core.Condition

Bases: `holoscan.core._core.Component`

Class representing a condition.

Attributes

`args`	The list of arguments associated with the component.
`description`	YAML formatted string describing the condition.
`fragment`	Fragment that the condition belongs to.
`id`	The identifier of the component.
`name`	The name of the condition.

spec	

Methods

`add_arg`(*args, **kwargs)	Overloaded function.
`initialize`(self)	initialization method for the condition.

`setup`(self, arg0)	setup method for the condition.

__init__(*self: holoscan.core._core.Condition*, **args*, ***kwargs*) None

Class representing a condition.

Can be initialized with any number of Python positional and keyword arguments.

If a *name* keyword argument is provided, it must be a *str* and will be used to set the name of the condition.

If a *fragment* keyword argument is provided, it must be of type *holoscan.core.Fragment* (or *holoscan.core.Application*). A single *Fragment* object can also be provided positionally instead.

Any other arguments will be cast from a Python argument type to a C++ *Arg* and stored in `self.args`. (For details on how the casting is done, see the *py_object_to_arg* utility).

Parameters

***args**

Positional arguments.

****kwargs**

Keyword arguments.

Raises

RuntimeError

If *name* kwarg is provided, but is not of *str* type. If multiple arguments of type *Fragment* are provided. If any other arguments cannot be converted to *Arg* type via *py_object_to_arg*.

add_arg(**args*, ***kwargs*)

Overloaded function.

1. add_arg(self: holoscan.core._core.ComponentBase, arg: holoscan.core._core.Arg) -> None

Add an argument to the component.

2. add_arg(self: holoscan.core._core.ComponentBase, arg: holoscan.core._core.ArgList) -> None

Add a list of arguments to the component.

property args

The list of arguments associated with the component.

Returns

arglist

property description

YAML formatted string describing the condition.

property fragment

Fragment that the condition belongs to.

Returns

name

property id

The identifier of the component.

The identifier is initially set to `-1`, and will become a valid value when the component is initialized.

With the default executor (*holoscan.gxf.GXFExecutor*), the identifier is set to the GXF component ID.

Returns

id

initialize(*self: holoscan.core._core.Condition*) None

initialization method for the condition.

property name

The name of the condition.

Returns

name

setup(*self: holoscan.core._core.Condition*, *arg0: holoscan.core._core.ComponentSpec*) None

setup method for the condition.

property spec

class holoscan.core.ConditionType

Bases: pybind11_builtins.pybind11_object

Enum class for Condition types.

Members:

NONE

MESSAGE_AVAILABLE

DOWNSTREAM_MESSAGE_AFFORDABLE

COUNT

BOOLEAN

Attributes

name	

value	

BOOLEAN = <*ConditionType.BOOLEAN: 4*>

COUNT = <*ConditionType.COUNT: 3*>

DOWNSTREAM_MESSAGE_AFFORDABLE = <*ConditionType.DOWNSTREAM_MESSAGE_AFFORDABLE: 2*>

MESSAGE_AVAILABLE = <*ConditionType.MESSAGE_AVAILABLE: 1*>

NONE = <*ConditionType.NONE: 0*>

__init__(*self: holoscan.core._core.ConditionType*, *value: int*) None

property name

property value

class holoscan.core.Config

Bases: pybind11_builtins.pybind11_object

Configuration class.

Represents configuration parameters as read from a YAML file.

Attributes

config_file	The configuration file (in YAML format) associated with the Config object.
prefix	TODO

__init__(*self: holoscan.core._core.Config*, *config_file: str*, *prefix: str = ''*) None

Configuration class.

Represents configuration parameters as read from a YAML file.

Parameters

config_file

The path to the configuration file (in YAML format).

prefix

TODO

property config_file

The configuration file (in YAML format) associated with the Config object.

property prefix

TODO

class holoscan.core.DLDevice

Bases: pybind11_builtins.pybind11_object

DLDevice class.

Attributes

device_id	The device id (int).
device_type	The device type (*DLDeviceType*).

__init__(*self: holoscan.core._core.DLDevice*, *arg0: holoscan.core._core.DLDeviceType*, *arg1: int*) None

property device_id

The device id (int).

property device_type

The device type (*DLDeviceType*).

The following device types are supported:

- *DLDeviceType.DLCPU*: system memory (kDLCPU)
- *DLDeviceType.DLCUDA*: CUDA GPU memory (kDLCUDA)
- *DLDeviceType.DLCUDAHost*: CUDA pinned memory (kDLCUDAHost)
- *DLDeviceType.DLCUDAManaged*: CUDA managed memory (kDLCUDAManaged)

class holoscan.core.DLDeviceType

Bases: pybind11_builtins.pybind11_object

Members:

DLCPU

DLCUDA

DLCUDAHOST

DLCUDAMANAGED

Attributes

name	

value	

DLCPU = *<DLDeviceType.DLCPU: 1>*

DLCUDA = *<DLDeviceType.DLCUDA: 2>*

DLCUDAHOST = *<DLDeviceType.DLCUDAHOST: 3>*

DLCUDAMANAGED = *<DLDeviceType.DLCUDAMANAGED: 13>*

__init__(*self: holoscan.core._core.DLDeviceType*, *value: int*) None

property name

property value

class holoscan.core.DataFlowMetric

Bases: pybind11_builtins.pybind11_object

Enum class for DataFlowMetric type.

Members:

MAX_MESSAGE_ID

MIN_MESSAGE_ID

MAX_E2E_LATENCY

AVG_E2E_LATENCY

MIN_E2E_LATENCY

NUM_SRC_MESSAGES

NUM_DST_MESSAGES

Attributes

name	

value	

AVG_E2E_LATENCY = *<DataFlowMetric.AVG_E2E_LATENCY: 3>*

MAX_E2E_LATENCY = *<DataFlowMetric.MAX_E2E_LATENCY: 2>*

MAX_MESSAGE_ID = *<DataFlowMetric.MAX_MESSAGE_ID: 0>*

MIN_E2E_LATENCY = *<DataFlowMetric.MIN_E2E_LATENCY: 4>*

MIN_MESSAGE_ID = <*DataFlowMetric.MIN_MESSAGE_ID: 1*>

NUM_DST_MESSAGES = <*DataFlowMetric.NUM_DST_MESSAGES: 6*>

NUM_SRC_MESSAGES = <*DataFlowMetric.NUM_SRC_MESSAGES: 5*>

__init__(*self: holoscan.core._core.DataFlowMetric*, *value: int*) None

property name

property value

class holoscan.core.DataFlowTracker

Bases: pybind11_builtins.pybind11_object

Data Flow Tracker class.

The DataFlowTracker class is used to track the data flow metrics for different paths between the root and leaf operators. This class is used by developers to get data flow metrics either during the execution of the application and/or as a summary after the application ends.

Methods

enable_logging(self[, filename, ...])	Enable logging of frames at the end of the every execution of a leaf Operator.
end_logging(self)	Write out any remaining messages from the log buffer and close the file
get_metric(*args, **kwargs)	Overloaded function.

`get_num_paths`(self)	The number of tracked paths
`get_path_strings`(self)	Return an array of strings which are path names.
`print`(self)	Print the result of the data flow tracking in pretty-printed format to the standard output
`set_discard_last_messages`(self, arg0)	Set the number of messages to discard at the end of the execution.
`set_skip_latencies`(self, arg0)	Set the threshold latency for which the end-to-end latency calculations will be done.
`set_skip_starting_messages`(self, arg0)	Set the number of messages to skip at the beginning of the execution.

__init__(*self:* *holoscan.core._core.DataFlowTracker*) None

Data Flow Tracker class.

The DataFlowTracker class is used to track the data flow metrics for different paths between the root and leaf operators. This class is used by developers to get data flow metrics either during the execution of the application and/or as a summary after the application ends.

enable_logging(*self: holoscan.core._core.DataFlowTracker*, *filename: str = 'logger.log'*, *num_buffered_messages: int = 100*) None

Enable logging of frames at the end of the every execution of a leaf Operator.

A path consisting of an array of tuples in the form of (an Operator name, message receive timestamp, message publish timestamp) is logged in a file. The logging does not take into account the number of message to skip or discard or the threshold latency.

This function buffers a number of lines set by *num_buffered_messages* before flushing the buffer to the log file.

Parameters

filename

The name of the log file.

num_buffered_messages

The number of messages to be buffered before flushing the buffer to the log file.

end_logging(*self: holoscan.core._core.DataFlowTracker*) None

Write out any remaining messages from the log buffer and close the file

get_metric(**args*, ***kwargs*)

Overloaded function.

1. get_metric(self: holoscan.core._core.DataFlowTracker, pathstring: str, metric: holoscan.core._core.DataFlowMetric) -> float

Return the value of a metric for a given path.

If *metric* is DataFlowMetric::NUM_SRC_MESSAGES, then the function returns -1.

Parameters

pathstring

The path name string for which the metric is being queried

metric

The metric to be queried.

Returns

val

The value of the metric for the given path

Notes

There is also an overloaded version of this function that takes only the *metric* argument.

2. get_metric(self: holoscan.core._core.DataFlowTracker, metric: holoscan.core._core.DataFlowMetric = <DataFlowMetric.NUM_SRC_MESSAGES: 5>) -> Dict[str, int]

get_num_paths(*self: holoscan.core._core.DataFlowTracker*) int

The number of tracked paths

Returns

num_paths

The number of tracked paths

get_path_strings(*self: holoscan.core._core.DataFlowTracker*) List[str]

Return an array of strings which are path names. Each path name is a comma-separated list of Operator names in a path. The paths are agnostic to the edges between two Operators.

Returns

paths

A list of the path names.

print(*self: holoscan.core._core.DataFlowTracker*) None

Print the result of the data flow tracking in pretty-printed format to the standard output

set_discard_last_messages(*self: holoscan.core._core.DataFlowTracker*, *arg0: int*) None

Set the number of messages to discard at the end of the execution.

This does not affect the log file or the number of source messages metric.

Parameters

num

The number of messages to discard.

set_skip_latencies(*self: holoscan.core._core.DataFlowTracker*, *arg0: int*) None

Set the threshold latency for which the end-to-end latency calculations will be done. Any latency strictly less than the threshold latency will be ignored.

This does not affect the log file or the number of source messages metric.

Parameters

threshold

The threshold latency in milliseconds.

set_skip_starting_messages(*self: holoscan.core._core.DataFlowTracker*, *arg0: int*) None

Set the number of messages to skip at the beginning of the execution.

This does not affect the log file or the number of source messages metric.

Parameters

num

The number of messages to skip.

class holoscan.core.ExecutionContext

Bases: `pybind11_builtins.pybind11_object`

Class representing an execution context.

__init__(**args*, ***kwargs*)

class holoscan.core.Executor

Bases: `pybind11_builtins.pybind11_object`

Executor class.

Attributes

`context`	The corresponding GXF context.
`context_uint64`	The corresponding GXF context represented as a 64-bit unsigned integer address
`fragment`	The fragment that the executor belongs to.

Methods

`run`(self, arg0)	Method that can be called to run the executor.

__init__(*self: holoscan.core._core.Executor*, *fragment: holoscan::Fragment*) None

Executor class.

Parameters

fragment

The fragment that the executor is associated with.

property context

The corresponding GXF context. This will be an opaque PyCapsule object.

property context_uint64

The corresponding GXF context represented as a 64-bit unsigned integer address

property fragment

The fragment that the executor belongs to.

Returns

name

run(*self: holoscan.core._core.Executor*, *arg0: holoscan.graphs._graphs.OperatorGraph*) None

Method that can be called to run the executor.

class holoscan.core.Fragment(*app=None*, *name=''*, **args*, ***kwargs*)

Bases: holoscan.core._core.Fragment

Fragment class.

Attributes

application	The application associated with the fragment.
executor	Get the executor associated with the fragment.
graph	Get the computation graph (Graph node is an Operator) associated with the fragment.

`name`	The fragment's name.

Methods

`add_flow`(*args, **kwargs)	Overloaded function.
`add_operator`(self, op)	Add an operator to the fragment.
`compose`(self)	The compose method of the Fragment.
`config`(*args, **kwargs)	Overloaded function.
`config_keys`(self)	The set of keys present in the fragment's configuration file.
`from_config`(self, key)	Retrieve parameters from the associated configuration.
`kwargs`(self, k	Retrieve a dictionary parameters from the associated configuration.

ey)	
network_context (*args, **kwargs)	Overloaded function.
run (self)	The run method of the Fragment.
run_async ()	Run the fragment asynchronously.
scheduler (*args, **kwargs)	Overloaded function.
track (self, num_start_messages_to_skip, ...)	The track method of the application.

__init__(*self: holoscan.core._core.Fragment*, *arg0: object*) None

Fragment class.

add_flow(**args*, ***kwargs*)

Overloaded function.

1. add_flow(self: holoscan.core._core.Fragment, upstream_op: holoscan.core._core.Operator, downstream_op: holoscan.core._core.Operator) -> None

2. add_flow(self: holoscan.core._core.Fragment, upstream_op: holoscan.core._core.Operator, downstream_op: holoscan.core._core.Operator, port_pairs: Set[Tuple[str, str]]) -> None

Connect two operators associated with the fragment.

Parameters

upstream_op

Source operator.

downstream_op

Destination operator.

port_pairs

Sequence of ports to connect. The first element of each 2-tuple is a port from *upstream_op* while the second element is the port of *downstream_op* to which it connects.

Notes

This is an overloaded function. Additional variants exist:

1.) For the Application class there is a variant where the first two arguments are of type *holoscan.core.Fragment* instead of *holoscan.core.Operator*. This variant is used in building multi-fragment applications. 2.) There are also variants that omit the *port_pairs* argument that are applicable when there is only a single output on the upstream operator/fragment and a single input on the downstream operator/fragment.

add_operator(*self: holoscan.core._core.Fragment*, *op: holoscan.core._core.Operator*) None

Add an operator to the fragment.

Parameters

op

The operator to add.

property application

The application associated with the fragment.

Returns

app

compose(*self: holoscan.core._core.Fragment*) None

The compose method of the Fragment.

This method should be called after *config*, but before *run* in order to compose the computation graph.

config(**args*, ***kwargs*)

Overloaded function.

1. config(self: holoscan.core._core.Fragment, config_file: str, prefix: str = '') -> None

Configuration class.

Represents configuration parameters as read from a YAML file.

Parameters

config

The path to the configuration file (in YAML format) or a *holoscan.core.Config* object.

prefix

Prefix path for the` config` file. Only available in the overloaded variant that takes a string for *config*.

2. config(self: holoscan.core._core.Fragment, arg0: holoscan.core._core.Config) -> None
3. config(self: holoscan.core._core.Fragment) -> holoscan.core._core.Config

config_keys(*self: holoscan.core._core.Fragment*) Set[str]

The set of keys present in the fragment's configuration file.

property executor

Get the executor associated with the fragment.

from_config(*self: holoscan.core._core.Fragment*, *key: str*) object

Retrieve parameters from the associated configuration.

Parameters

key

The key within the configuration file to retrieve. This can also be a specific component of the parameter via syntax *'key.sub_key'*.

Returns

args

An argument list associated with the key.

property graph

Get the computation graph (Graph node is an Operator) associated with the fragment.

kwargs(*self: holoscan.core._core.Fragment*, *key: str*) dict

Retrieve a dictionary parameters from the associated configuration.

Parameters

key

The key within the configuration file to retrieve. This can also be a specific component of the parameter via syntax *'key.sub_key'*.

Returns

kwargs

A Python dict containing the parameters in the configuration file under the specified key.

property name

The fragment's name.

Returns

name

network_context(**args*, ***kwargs*)

Overloaded function.

1. network_context(self: holoscan.core._core.Fragment, network_context: holoscan.core._core.NetworkContext) -> None

Assign a network context to the Fragment

Parameters

network_context

A network_context class instance to be used by the underlying GXF executor. If unspecified, no network context will be used.

2. network_context(self: holoscan.core._core.Fragment) -> holoscan.core._core.NetworkContext
Get the network context to be used by the Fragment

run(*self: holoscan.core._core.Fragment*) None

The run method of the Fragment.

This method runs the computation. It must have first been initialized via *config* and *compose*.

run_async()

Run the fragment asynchronously.

This method is a convenience method that creates a thread pool with one thread and runs the fragment in that thread. The thread pool is created using *concurrent.futures.ThreadPoolExecutor*.

Returns

future : concurrent.futures.Future

scheduler(**args*, ***kwargs*)

Overloaded function.

1. scheduler(self: holoscan.core._core.Fragment, scheduler: holoscan.core._core.Scheduler) -> None

Assign a scheduler to the Fragment.

Parameters

scheduler

A scheduler class instance to be used by the underlying GXF executor. If unspecified, the default is a *holoscan.gxf.GreedyScheduler*.

2. scheduler(self: holoscan.core._core.Fragment) -> holoscan.core._core.Scheduler
Get the scheduler to be used by the Fragment.

track(*self: holoscan.core._core.Fragment*, *num_start_messages_to_skip: int = 10*, *num_last_messages_to_discard: int = 10*, *latency_threshold: int = 0*)
holoscan::DataFlowTracker

The track method of the application.

This method enables data frame flow tracking and returns a DataFlowTracker object which can be used to display metrics data for profiling an application.

Parameters

num_start_messages_to_skip

The number of messages to skip at the beginning.

num_last_messages_to_discard

The number of messages to discard at the end.

latency_threshold

The minimum end-to-end latency in milliseconds to account for in the end-to-end latency metric calculations

class holoscan.core.FragmentGraph

Bases: `pybind11_builtins.pybind11_object`

Abstract base class for all graphs

__init__(**args*, ***kwargs*)

holoscan.core.Graph

alias of `holoscan.graphs._graphs.OperatorGraph`

class holoscan.core.IOSpec

Bases: `pybind11_builtins.pybind11_object`

I/O specification class.

Attributes

`conditions`	List of Condition objects associated with this I/O specification.
`connector_type`	The receiver or transmitter type of the I/O specification class.
`io_type`	The type (input or output) of the I/O specification class.
`name`	The name of the I/O specification class.

Methods

condition(self, arg0, **kwargs)	Add a condition to this input/output.
connector(*args, **kwargs)	Overloaded function.

ConnectorType	
IOType	

class ConnectorType

Bases: pybind11_builtins.pybind11_object

Enum representing the receiver type (for input specs) or transmitter type (for output specs).

Members:

DEFAULT

DOUBLE_BUFFER

UCX

Attributes

name	

value	

DEFAULT = *<ConnectorType.DEFAULT: 0>*

DOUBLE_BUFFER = *<ConnectorType.DOUBLE_BUFFER: 1>*

UCX = *<ConnectorType.UCX: 2>*

__init__(*self: holoscan.core._core.IOSpec.ConnectorType*, *value: int*) None

property name

property value

class IOType

Bases: pybind11_builtins.pybind11_object

Enum representing the I/O specification type (input or output).

Members:

INPUT

OUTPUT

Attributes

name	

value	

INPUT = *<IOType.INPUT: 0>*

OUTPUT = *<IOType.OUTPUT: 1>*

__init__(*self: holoscan.core._core.IOSpec.IOType*, *value: int*) None

property name

property value

__init__(*self: holoscan.core._core.IOSpec*, *op_spec: holoscan::OperatorSpec*, *name: str*, *io_type: holoscan.core._core.IOSpec.IOType*) None

I/O specification class.

Parameters

op_spec

Operator specification class of the associated operator.

name

The name of the IOSpec object.

io_type

Enum indicating whether this is an input or output specification.

condition(*self: holoscan.core._core.IOSpec*, *arg0: holoscan.core._core.ConditionType*, ***kwargs*) holoscan.core._core.IOSpec

Add a condition to this input/output.

The following ConditionTypes are supported:

- *ConditionType.NONE*
- *ConditionType.MESSAGE_AVAILABLE*
- *ConditionType.DOWNSTREAM_MESSAGE_AFFORDABLE*
- *ConditionType.COUNT*
- *ConditionType.BOOLEAN*

Parameters

kind

The type of the condition.

****kwargs**

Python keyword arguments that will be cast to an *ArgList* associated with the condition.

Returns

obj

The self object.

property conditions

List of Condition objects associated with this I/O specification.

Returns

condition

connector(**args*, ***kwargs*)

Overloaded function.

1. connector(self: holoscan.core._core.IOSpec, arg0: holoscan.core._core.IOSpec.ConnectorType, **kwargs) -> holoscan.core._core.IOSpec

Add a connector (transmitter or receiver) to this input/output.

The following ConditionTypes are supported:

- *IOSpec.ConnectorType.DEFAULT*
- *IOSpec.ConnectorType.DOUBLE_BUFFER*
- *IOSpec.ConnectorType.UCX*

If this method is not been called, the IOSpec's *connector_type* will be *ConnectorType.DEFAULT* which will result in a DoubleBuffered receiver or or transmitter being used (or their annotated variant if flow tracking is enabled).

Parameters

kind

The type of the connector. For example for type *IOSpec.ConnectorType.DOUBLE_BUFFER*, a *holoscan.resources.DoubleBufferReceiver* will be used for an input port and a *holoscan.resources.DoubleBufferTransmitter* will be used for an output port.

****kwargs**

Python keyword arguments that will be cast to an *ArgList* associated with the resource (connector).

Returns

obj

The self object.

Notes

This is an overloaded function. Additional variants exist:

1.) A variant with no arguments will just return the *holoscan.core.Resource* corresponding to the transmitter or receiver used by this *IOSpec* object. If None was explicitly set, it will return `None`.

2.) A variant that takes a single *holoscan.core.Resource* corresponding to a transmitter or receiver as an argument. This sets the transmitter or receiver used by the *IOSpec* object.

2. connector(self: holoscan.core._core.IOSpec) -> holoscan.core._core.Resource

3. connector(self: holoscan.core._core.IOSpec, arg0: holoscan.core._core.Resource) -> None

property connector_type

The receiver or transmitter type of the I/O specification class.

Returns

connector_type

property io_type

The type (input or output) of the I/O specification class.

Returns

io_type

property name

The name of the I/O specification class.

Returns

name

class holoscan.core.InputContext

Bases: pybind11_builtins.pybind11_object

Class representing an input context.

Methods

receive (self, name)	

__init__(**args*, ***kwargs*)

receive(*self: holoscan.core._core.InputContext*, *name: str*) None

class holoscan.core.Message

Bases: pybind11_builtins.pybind11_object

Class representing a message.

A message is a data structure that is used to pass data between operators. It wraps a std::any object and provides a type-safe interface to access the data.

This class is used by the *holoscan::gxf::GXFWrapper* to support the Holoscan native operator. The *holoscan::gxf::GXFWrapper* will hold the object of this class and

delegate the message to the Holoscan native operator.

__init__(**args*, ***kwargs*)

class holoscan.core.NetworkContext

Bases: holoscan.core._core.Component

Class representing a network context.

Attributes

args	The list of arguments associated with the component.
description	YAML formatted string describing the component.
fragment	Fragment that the network context belongs to.
id	The identifier of the component.
name	The name of the network context.

spec	

Methods

add_arg(*args, **kwargs)	Overloaded function.
initialize(self)	initialization method for the network context.

setup(self, arg0)	setup method for the network context.

__init__(*self: holoscan.core._core.NetworkContext*, **args*, ***kwargs*) None

Class representing a network context.

Parameters

***args**

Positional arguments.

****kwargs**

Keyword arguments.

Raises

RuntimeError

If *name* kwarg is provided, but is not of *str* type. If multiple arguments of type *Fragment* are provided. If any other arguments cannot be converted to *Arg* type via *py_object_to_arg*.

add_arg(**args*, ***kwargs*)

Overloaded function.

1. add_arg(self: holoscan.core._core.ComponentBase, arg: holoscan.core._core.Arg) -> None

Add an argument to the component.

2. add_arg(self: holoscan.core._core.ComponentBase, arg: holoscan.core._core.ArgList) -> None

Add a list of arguments to the component.

property args

The list of arguments associated with the component.

Returns

arglist

property description

YAML formatted string describing the component.

property fragment

Fragment that the network context belongs to.

Returns

name

property id

The identifier of the component.

The identifier is initially set to `-1`, and will become a valid value when the component is initialized.

With the default executor (*holoscan.gxf.GXFExecutor*), the identifier is set to the GXF component ID.

Returns

id

initialize(*self:* *holoscan.core._core.NetworkContext*) None

initialization method for the network context.

property name

The name of the network context.

Returns

name

setup(*self: holoscan.core._core.NetworkContext*, *arg0: holoscan.core._core.ComponentSpec*) → None

setup method for the network context.

property spec

class holoscan.core.Operator(*fragment*, **args*, ***kwargs*)

Bases: holoscan.core._core.Operator

Operator class.

Can be initialized with any number of Python positional and keyword arguments.

If a *name* keyword argument is provided, it must be a *str* and will be used to set the name of the operator.

Condition classes will be added to `self.conditions`, *Resource* classes will be added to `self.resources`, and any other arguments will be cast from a Python argument type to a C++ *Arg* and stored in `self.args`. (For details on how the casting is done, see the *py_object_to_arg* utility). When a Condition or Resource is provided via a kwarg, it's name will be automatically be updated to the name of the kwarg.

Parameters

fragment

The *holoscan.core.Fragment* (or *holoscan.core.Application*) to which this Operator will belong.

***args**

Positional arguments.

****kwargs**

Keyword arguments.

Raises

RuntimeError

If *name* kwarg is provided, but is not of *str* type. If multiple arguments of type *Fragment* are provided. If any other arguments cannot be converted to *Arg* type via *py_object_to_arg*.

Attributes

`args`	The list of arguments associated with the component.
`conditions`	Conditions associated with the operator.
`description`	YAML formatted string describing the operator.
`fragment`	The fragment (`holoscan.core.Fragment`) that the operator belongs to.
`id`	The identifier of the component.
`name`	The name of the operator.
`operator_type`	The operator type.
`resources`	Resources associated with the operator.
`spec`	The operator spec (`holoscan.core.OperatorSpec`) associated with the operator.

Methods

`add_arg` (*args, **kwargs)	Overloaded function.

`compute`(op_input, op_output, context)	Default implementation of compute
`initialize`()	Default implementation of initialize
`setup`(spec)	Default implementation of setup method.
`start`()	Default implementation of start
`stop`()	Default implementation of stop

OperatorType	

class OperatorType

Bases: `pybind11_builtins.pybind11_object`

Enum class for operator types used by the executor.

- NATIVE: Native operator.
- GXF: GXF operator.
- VIRTUAL: Virtual operator. (for internal use, not intended for use by application authors)

Members:

NATIVE

GXF

VIRTUAL

Attributes

name	

value	

GXF = <*OperatorType.GXF: 1*>

NATIVE = <*OperatorType.NATIVE: 0*>

VIRTUAL = <*OperatorType.VIRTUAL: 2*>

__init__(*self: holoscan.core._core.Operator.OperatorType*, *value: int*) None

property name

property value

__init__(*self: holoscan.core._core.Operator*, *arg0: object*, *arg1: holoscan::Fragment*, **args*, ***kwargs*) None

Operator class.

Can be initialized with any number of Python positional and keyword arguments.

If a *name* keyword argument is provided, it must be a *str* and will be used to set the name of the operator.

Condition classes will be added to `self.conditions`, *Resource* classes will be added to `self.resources`, and any other arguments will be cast from a Python argument type to a C++ *Arg* and stored in `self.args`. (For details on how the casting is done, see the *py_object_to_arg* utility). When a Condition or Resource

is provided via a kwarg, it's name will be automatically be updated to the name of the kwarg.

Parameters

fragment

The *holoscan.core.Fragment* (or *holoscan.core.Application*) to which this Operator will belong.

***args**

Positional arguments.

****kwargs**

Keyword arguments.

Raises

RuntimeError

If *name* kwarg is provided, but is not of *str* type. If multiple arguments of type *Fragment* are provided. If any other arguments cannot be converted to *Arg* type via *py_object_to_arg*.

add_arg(**args*, ***kwargs*)

Overloaded function.

1. add_arg(self: holoscan.core._core.Operator, arg: holoscan.core._core.Arg) -> None

Add an argument to the component.

2. add_arg(self: holoscan.core._core.Operator, arg: holoscan.core._core.ArgList) -> None

Add a list of arguments to the component.

3. add_arg(self: holoscan.core._core.Operator, **kwargs) -> None

Add arguments to the component via Python kwargs.

4. add_arg(self: holoscan.core._core.Operator, arg: holoscan.core._core.Condition) -> None

5. add_arg(self: holoscan.core._core.Operator, arg: holoscan.core._core.Resource) -> None

Add a condition or resource to the Operator.

This can be used to add a condition or resource to an operator after it has already been constructed.

Parameters

arg

The condition or resource to add.

property args

The list of arguments associated with the component.

Returns

arglist

compute(*op_input*, *op_output*, *context*)

Default implementation of compute

property conditions

Conditions associated with the operator.

property description

YAML formatted string describing the operator.

property fragment

The fragment (`holoscan.core.Fragment`) that the operator belongs to.

property id

The identifier of the component.

The identifier is initially set to `-1`, and will become a valid value when the component is initialized.

With the default executor (*holoscan.gxf.GXFExecutor*), the identifier is set to the GXF component ID.

Returns

id

initialize()

Default implementation of initialize

property name

The name of the operator.

property operator_type

The operator type.

holoscan.core.Operator.OperatorType enum representing the type of the operator. The two types currently implemented are native and GXF.

property resources

Resources associated with the operator.

setup(*spec: holoscan.core._core.PyOperatorSpec*)

Default implementation of setup method.

property spec

The operator spec (`holoscan.core.OperatorSpec`) associated with the operator.

start()

Default implementation of start

stop()

Default implementation of stop

class holoscan.core.OperatorGraph

Bases: pybind11_builtins.pybind11_object

Abstract base class for all graphs

__init__(**args*, ***kwargs*)

holoscan.core.OperatorSpec

alias of holoscan.core._core.PyOperatorSpec

class holoscan.core.OutputContext

Bases: pybind11_builtins.pybind11_object

Class representing an output context.

Methods

emit (self, data[, name])	

OutputType	

class OutputType

Bases: pybind11_builtins.pybind11_object

Members:

SHARED_POINTER

GXF_ENTITY

Attributes

name	

value	

GXF_ENTITY = <*OutputType.GXF_ENTITY: 1*>

SHARED_POINTER = <*OutputType.SHARED_POINTER: 0*>

__init__(*self: holoscan.core._core.OutputContext.OutputType*, *value: int*) None

property name

property value

__init__(**args*, ***kwargs*)

emit(*self: holoscan.core._core.OutputContext*, *data: object*, *name: str = ''*) None

class holoscan.core.ParameterFlag

Bases: pybind11_builtins.pybind11_object

Enum class for parameter flags.

The following flags are supported: - *NONE*: The parameter is mendatory and static. It cannot be changed at runtime. - *OPTIONAL*: The parameter is optional and might not be available at runtime. - *DYNAMIC*: The parameter is dynamic and might change at runtime.

Members:

NONE

OPTIONAL

DYNAMIC

Attributes

name	

value	

DYNAMIC = *<ParameterFlag.DYNAMIC: 2>*

NONE = *<ParameterFlag.NONE: 0>*

OPTIONAL = *<ParameterFlag.OPTIONAL: 1>*

__init__(*self: holoscan.core._core.ParameterFlag*, *value: int*) None

property name

property value

class holoscan.core.Resource

Bases: holoscan.core._core.Component

Class representing a resource.

Attributes

args	The list of arguments associated with the component.
description	YAML formatted string describing the resource.
fragment	Fragment that the resource belongs to.
id	The identifier of the component.
name	The name of the resource.

spec	

Methods

`add_arg`(*args, **kwargs)	Overloaded function.
`initialize`(self)	initialization method for the resource.
`setup`(self, arg0)	setup method for the resource.

__init__(*self: holoscan.core._core.Resource*, **args*, ***kwargs*) None

Class representing a resource.

Can be initialized with any number of Python positional and keyword arguments.

If a *name* keyword argument is provided, it must be a *str* and will be used to set the name of the resource.

If a *fragment* keyword argument is provided, it must be of type *holoscan.core.Fragment* (or *holoscan.core.Application*). A single *Fragment* object can also be provided positionally instead.

Any other arguments will be cast from a Python argument type to a C++ *Arg* and stored in `self.args`. (For details on how the casting is done, see the *py_object_to_arg* utility).

Parameters

***args**

Positional arguments.

****kwargs**

Keyword arguments.

Raises

RuntimeError

If *name* kwarg is provided, but is not of *str* type. If multiple arguments of type *Fragment* are provided. If any other arguments cannot be converted to *Arg* type via *py_object_to_arg*.

add_arg(**args*, ***kwargs*)

Overloaded function.

1. add_arg(self: holoscan.core._core.ComponentBase, arg: holoscan.core._core.Arg) -> None

Add an argument to the component.

2. add_arg(self: holoscan.core._core.ComponentBase, arg: holoscan.core._core.ArgList) -> None

Add a list of arguments to the component.

property args

The list of arguments associated with the component.

Returns

arglist

property description

YAML formatted string describing the resource.

property fragment

Fragment that the resource belongs to.

Returns

name

property id

The identifier of the component.

The identifier is initially set to `-1`, and will become a valid value when the component is initialized.

With the default executor (*holoscan.gxf.GXFExecutor*), the identifier is set to the GXF component ID.

Returns

id

initialize(*self: holoscan.core._core.Resource*) None

initialization method for the resource.

property name

The name of the resource.

Returns

name

setup(*self: holoscan.core._core.Resource*, *arg0: holoscan.core._core.ComponentSpec*) None

setup method for the resource.

property spec

class holoscan.core.Scheduler

Bases: `holoscan.core._core.Component`

Class representing a scheduler.

Attributes

`args`	The list of arguments associated with the component.

description	YAML formatted string describing the component.
fragment	Fragment that the scheduler belongs to.
id	The identifier of the component.
name	The name of the scheduler.

spec	

Methods

add_arg(*args, **kwargs)	Overloaded function.
initialize(self)	initialization method for the scheduler.
setup(self, arg0)	setup method for the scheduler.

__init__(*self: holoscan.core._core.Scheduler*, **args*, ***kwargs*) None

Class representing a scheduler.

Can be initialized with any number of Python positional and keyword arguments.

If a *name* keyword argument is provided, it must be a *str* and will be used to set the name of the scheduler.

If a *fragment* keyword argument is provided, it must be of type *holoscan.core.Fragment* (or *holoscan.core.Application*). A single *Fragment* object can also be provided positionally instead.

Any other arguments will be cast from a Python argument type to a C++ *Arg* and stored in `self.args`. (For details on how the casting is done, see the *py_object_to_arg* utility).

Parameters

***args**

Positional arguments.

****kwargs**

Keyword arguments.

Raises

RuntimeError

If *name* kwarg is provided, but is not of *str* type. If multiple arguments of type *Fragment* are provided. If any other arguments cannot be converted to *Arg* type via *py_object_to_arg*.

add_arg(**args*, ***kwargs*)

Overloaded function.

1. add_arg(self: holoscan.core._core.ComponentBase, arg: holoscan.core._core.Arg) -> None

Add an argument to the component.

2. add_arg(self: holoscan.core._core.ComponentBase, arg: holoscan.core._core.ArgList) -> None

Add a list of arguments to the component.

property args

The list of arguments associated with the component.

Returns

arglist

property description

YAML formatted string describing the component.

property fragment

Fragment that the scheduler belongs to.

Returns

name

property id

The identifier of the component.

The identifier is initially set to `-1`, and will become a valid value when the component is initialized.

With the default executor (*holoscan.gxf.GXFExecutor*), the identifier is set to the GXF component ID.

Returns

id

initialize(*self: holoscan.core._core.Scheduler*) None

initialization method for the scheduler.

property name

The name of the scheduler.

Returns

name

setup(*self: holoscan.core._core.Scheduler*, *arg0: holoscan.core._core.ComponentSpec*) None

setup method for the scheduler.

property spec

holoscan.core.Tensor

alias of `holoscan.core._core.PyTensor`

class holoscan.core.Tracker(*app*, *, *filename=None*, *num_buffered_messages=100*, *num_start_messages_to_skip=10*, *num_last_messages_to_discard=10*, *latency_threshold=0*)

Bases: `object`

Context manager to add data flow tracking to an application.

__init__(*app*, *, *filename=None*, *num_buffered_messages=100*, *num_start_messages_to_skip=10*, *num_last_messages_to_discard=10*, *latency_threshold=0*)

Parameters

app

on which flow tracking should be applied.

filename

If none, logging to file will be disabled. Otherwise, logging will write to the specified file.

num_buffered_messages

Controls the number of messages buffered between file writing when *filename* is not `None`.

num_start_messages_to_skip

The number of messages to skip at the beginning of the execution. This does not affect the log file or the number of source messages metric.

num_last_messages_to_discard

The number of messages to discard at the end of the execution. This does not affect the log file or the number of source messages metric.

latency_threshold

The minimum end-to-end latency in milliseconds to account for in the end-to-end latency metric calculations.

holoscan.core.arg_to_py_object(*arg: holoscan.core._core.Arg*) object

Utility that converts an *Arg* to a corresponding Python object.

Parameters

arg

The argument to convert.

Returns

obj

Python object corresponding to the provided argument. For example, an argument of any integer type will become a Python *int* while *std::vector<double>* would become a list of Python floats.

holoscan.core.arglist_to_kwargs(*arglist: holoscan.core._core.ArgList*) dict

Utility that converts an *ArgList* to a Python kwargs dictionary.

Parameters

arglist

The argument list to convert.

Returns

kwargs

Python dictionary with keys matching the names of the arguments in *ArgList*. The values will be converted as for *arg_to_py_object*.

holoscan.core.kwargs_to_arglist(***kwargs*) holoscan.core._core.ArgList

Utility that converts a set of python keyword arguments to an *ArgList*.

Parameters

****kwargs**

The python keyword arguments to convert.

Returns

arglist

ArgList class corresponding to the provided keyword values. The argument names will match the keyword names. Values will be converted as for *py_object_to_arg*.

holoscan.core.py_object_to_arg(*obj: object*, *name: str = ''*) holoscan.core._core.Arg

Utility that converts a single python argument to a corresponding *Arg* type.

Parameters

value

The python value to convert.

Returns

obj

Arg class corresponding to the provided value. For example a Python float will become an *Arg* containing a C++ double while a list of Python ints would become an *Arg* corresponding to a `std::vector<uint64_t>`.

name

A name to assign to the argument.