



**holoscan.conditions**

This module provides a Python API to underlying C++ API Conditions.

<code>holoscan.conditions.AsyncCondition</code>	Asynchronous condition class.
<code>holoscan.conditions.BooleanCondition</code>	Boolean condition.
<code>holoscan.conditions.CountCondition</code>	Count condition.
<code>holoscan.conditions.DownstreamMessageAffordableCondition</code>	Condition that permits execution when the downstream operator can accept new messages.

holoscan.conditions.MessageAvailabilityCondition	Condition that permits execution when an upstream message is available.
holoscan.conditions.PeriodicCondition	Condition class to support periodic execution of operators.

*class* holoscan.conditions.AsynchronousCondition

Bases: holoscan.gxf.\_gxf.GXFCondition

Asynchronous condition class.

Used to control whether an entity is executed.

Attributes

args	The list of arguments associated with the component.
description	YAML formatted string describing the condition.
event_state	Event state property
fragment	Fragment that the condition belongs to.

<code>gxf_cid</code>	The GXF component ID.
<code>gxf_name</code>	The name of the component.
<code>gxf_context</code>	The GXF context of the component.
<code>gxf_entity_id</code>	The GXF entity ID.
<code>gxf_type_name</code>	The GXF type name of the condition.
<code>id</code>	The identifier of the component.
<code>name</code>	The name of the condition.

<b>spec</b>	
-------------	--

## Methods

<code>add_arg</code> (*args, **kwargs)	Overloaded function.
<code>gxf_initialize</code> (self)	Initialize the component.
<code>initialize</code> (self)	Initialize the component.

<pre> setu p (self, s pec) </pre>	Define the component specification.
-----------------------------------	-------------------------------------

`__init__(self: holoscan.conditions.\_conditions.AsynchronousCondition, fragment: holoscan.core.\_core.Fragment, name: str = 'noname_async_condition') None`

Asynchronous condition.

Parameters

**fragment**

The fragment the condition will be associated with

**name**

The name of the condition.

`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* event\_state

Event state property

- AsynchronousEventState.READY
- AsynchronousEventState.WAIT
- AsynchronousEventState.EVENT\_WAITING
- AsynchronousEventState.EVENT\_DONE
- AsynchronousEventState.EVENT\_NEVER

*property* fragment

Fragment that the condition belongs to.

Returns

**name**

*property* gxf\_cid

The GXF component ID.

*property* gxf\_cname

The name of the component.

*property* gxf\_context

The GXF context of the component.

*property* gxf\_eid

The GXF entity ID.

*gxf\_initialize(self: [holoscan.gxf.\\_gxf.GXFComponent](#))* None

Initialize the component.

*property* `gxf_typename`

The GXF type name of the condition.

Returns

`str`

The GXF type name of the condition

*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.gxf._gxf.GXFCondition)` None

Initialize the component.

*property* `name`

The name of the condition.

Returns

**name**

`setup(self: holoscan.conditions._conditions.AsynchronousCondition, spec: holoscan.core._core.ComponentSpec)` None

Define the component specification.

## Parameters

### **spec**

Component specification associated with the condition.

*property spec*

*class* holoscan.conditions.AsynchronousEventState

Bases: `pybind11_builtins.pybind11_object`

Members:

READY

WAIT

EVENT\_WAITING

EVENT\_DONE

EVENT\_NEVER

Attributes

name	
------	--

<b>value</b>	
--------------	--

EVENT\_DONE = <AsynchronousEventState.EVENT\_DONE: 3>

EVENT\_NEVER = <AsynchronousEventState.EVENT\_NEVER: 4>

EVENT\_WAITING = <AsynchronousEventState.EVENT\_WAITING: 2>

READY = <AsynchronousEventState.READY: 0>

WAIT = <AsynchronousEventState.WAIT: 1>

`__init__(self: holoscan.conditions.\_conditions.AsynchronousEventState, value: int) None`



*property name*

*property value*

class holoscan.conditions.BooleanCondition

Bases: `holoscan.gxf._gxf.GXFCondition`

Boolean condition.

Parameters

**fragment**

The fragment the condition will be associated with

**enable\_tick**

Boolean value for the condition.

**name**

The name of the 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.
gxf_cid	The GXF component ID.
gxf_name	The name of the component.

<code>gxf_context</code>	The GXF context of the component.
<code>gxf_entity_id</code>	The GXF entity ID.
<code>gxf_type_name</code>	The GXF type name of the condition.
<code>id</code>	The identifier of the component.
<code>name</code>	The name of the condition.

## spec

### Methods

<code>add_arg</code> (*args, **kwargs)	Overloaded function.
<code>check_tick_enabled</code> (self)	Check whether the condition is <code>True</code> .
<code>disable_tick</code> (self)	Set condition to <code>False</code> .
<code>enable_tick</code> (self)	Set condition to <code>True</code> .

<pre>gxf_initialize (self)</pre>	Initialize the component.
<pre>initialize (self)</pre>	Initialize the component.
<pre>setup (self, specification)</pre>	Define the component specification.

```
__init__(self: holoscan.conditions._conditions.BooleanCondition, fragment:
holoscan.core._core.Fragment, enable_tick: bool = True, name: str =
'noname_boolean_condition') None
```

Boolean condition.

Parameters

**fragment**

The fragment the condition will be associated with

**enable\_tick**

Boolean value for the condition.

**name**

The name of the condition.

```
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**

`check_tick_enabled(self: holoscan.conditions._conditions.BooleanCondition) bool`

Check whether the condition is `True`.

*property* description

YAML formatted string describing the condition.

`disable_tick(self: holoscan.conditions._conditions.BooleanCondition) None`

Set condition to `False`.

`enable_tick(self: holoscan.conditions._conditions.BooleanCondition) None`

Set condition to `True`.

*property* fragment

Fragment that the condition belongs to.

Returns

### **name**

*property* gxf\_cid

The GXF component ID.

*property* gxf\_cname

The name of the component.

*property* gxf\_context

The GXF context of the component.

*property* gxf\_eid

The GXF entity ID.

`gxf_initialize(self: holoscan.gxf.gxf.GXFComponent)` None

Initialize the component.

*property* gxf\_typename

The GXF type name of the condition.

Returns

str

The GXF type name of the condition

*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.gxf.gxf.GXFCondition)` None

Initialize the component.

*property* name

The name of the condition.

Returns

**name**

setup(self: [holoscan.conditions.\\_conditions.BooleanCondition](#), spec: [holoscan.core.\\_core.ComponentSpec](#)) None

Define the component specification.

Parameters

**spec**

Component specification associated with the condition.

*property spec*

*class* holoscan.conditions.CountCondition

Bases: [holoscan.gxf.\\_gxf.GXFCondition](#)

Count condition.

Parameters

**fragment**

The fragment the condition will be associated with

**count**

The execution count value used by the condition.

**name**

The name of the condition.

Attributes

[args](#)

The list of arguments associated with the component.

count	The execution count associated with the condition
description	YAML formatted string describing the condition.
fragment	Fragment that the condition belongs to.
gxf_cid	The GXF component ID.
gxf_name	The name of the component.
gxf_context	The GXF context of the component.
gxf_entity_id	The GXF entity ID.
gxf_type_name	The GXF type name of the condition.
id	The identifier of the component.
name	The name of the condition.

## spec

### Methods

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

<pre>gxf_initialize (self)</pre>	Initialize the component.
<pre>initialize (self)</pre>	Initialize the component.
<pre>setup (self, arg0)</pre>	Define the component specification.

```
__init__(self: holoscan.conditions._conditions.CountCondition, fragment:
holoscan.core._core.Fragment, count: int = 1, name: str = 'noname_count_condition')
None
```

Count condition.

Parameters

**fragment**

The fragment the condition will be associated with

**count**

The execution count value used by the condition.

**name**

The name of the condition.

```
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* count

The execution count associated with the condition

*property* description

YAML formatted string describing the condition.

*property* fragment

Fragment that the condition belongs to.

Returns

**name**

*property* gxf\_cid

The GXF component ID.

*property* gxf\_cname

The name of the component.

*property* gxf\_context

The GXF context of the component.

*property* gxf\_eid

The GXF entity ID.

`gxf_initialize(self: holoscan.gxf.\_gxf.GXFComponent)` None

Initialize the component.

*property* `gxf_typename`

The GXF type name of the condition.

Returns

str

The GXF type name of the condition

*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.gxf.\_gxf.GXFCondition)` None

Initialize the component.

*property* `name`

The name of the condition.

Returns

**name**

`setup(self: holoscan.conditions.\_conditions.CountCondition, arg0: holoscan.core.\_core.ComponentSpec)` None

Define the component specification.

Parameters

**spec**

Component specification associated with the condition.

*property spec*

*class* holoscan.conditions.DownstreamMessageAffordableCondition

Bases: holoscan.gxf.\_gxf.GXFCondition

Condition that permits execution when the downstream operator can accept new messages.

Satisfied when the receiver queue of any connected downstream operators has at least a certain number of elements free. The minimum number of messages that permits the execution of the entity is specified by *min\_size*. It can be used for operators to prevent operators from sending a message when the downstream operator is not ready to receive it.

Parameters

**fragment**

The fragment the condition will be associated with

**min\_size**

The minimum number of free slots present in the back buffer.

**name**

The name of the 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.
gxf_cid	The GXF component ID.
gxf_name	The name of the component.
gxf_context	The GXF context of the component.
gxf_entity_id	The GXF entity ID.
gxf_type_name	The GXF type name of the condition.
id	The identifier of the component.
min_size	The minimum number of free slots required for the downstream entity's back buffer.
name	The name of the condition.
transmitter	The transmitter associated with the condition.

<b>spec</b>	
-------------	--

## Methods

<pre>add_ arg (*args, **kwa rgs)</pre>	Overloaded function.
<pre>gxf_i nitiali ze (self)</pre>	Initialize the component.
<pre>initial ize (self)</pre>	Initialize the condition
<pre>setu p (self, s pec)</pre>	Define the component specification.

```
__init__(self: holoscan.conditions._conditions.DownstreamMessageAffordableCondition,
fragment: holoscan.core._core.Fragment, min_size: int = 1, name: str =
'noname_downstream_affordable_condition') None
```

Condition that permits execution when the downstream operator can accept new messages.

Satisfied when the receiver queue of any connected downstream operators has at least a certain number of elements free. The minimum number of messages that permits the execution of the entity is specified by *min\_size*. It can be used for operators to prevent operators from sending a message when the downstream operator is not ready to receive it.

Parameters

**fragment**

The fragment the condition will be associated with

**min\_size**

The minimum number of free slots present in the back buffer.

## **name**

The name of the condition.

`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 gxf\_cid*

The GXF component ID.

*property gxf\_cname*

The name of the component.

*property* gxf\_context

The GXF context of the component.

*property* gxf\_eid

The GXF entity ID.

`gxf_initialize(self: holoscan.gxf.gxf.GXFComponent)` None

Initialize the component.

*property* gxf\_typename

The GXF type name of the condition.

Returns

str

The GXF type name of the condition

*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.conditions.\_conditions.DownstreamMessageAffordableCondition)`  
None

Initialize the condition

This method is called only once when the condition is created for the first time, and uses a light-weight initialization.

*property* min\_size

The minimum number of free slots required for the downstream entity's back buffer.

*property* name

The name of the condition.

Returns

**name**

setup(*self*: [holoscan.conditions.\\_conditions.DownstreamMessageAffordableCondition](#),  
*spec*: [holoscan.core.\\_core.ComponentSpec](#)) None

Define the component specification.

Parameters

**spec**

Component specification associated with the condition.

*property* spec

*property* transmitter

The transmitter associated with the condition.

*class* holoscan.conditions.MessageAvailableCondition

Bases: [holoscan.gxf.\\_gxf.GXFCondition](#)

Condition that permits execution when an upstream message is available.

Satisfied when the associated receiver queue has at least a certain number of elements. The receiver is specified using the receiver parameter of the scheduling term. The minimum number of messages that permits the execution of the entity is specified by *min\_size*. An optional parameter for this scheduling term is



*front\_stage\_max\_size*, the maximum front stage message count. If this parameter is set, the scheduling term will only allow execution if the number of messages in the queue does not exceed this count. It can be used for operators which do not consume all messages from the queue.

## Parameters

### **fragment**

The fragment the condition will be associated with

### **min\_size**

The total number of messages over a set of input channels needed to permit execution.

### **front\_stage\_max\_size**

Threshold for the number of front stage messages. Execution is only allowed if the number of front stage messages does not exceed this count.

### **name**

The name of the 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.
front_stage_max_size	Threshold for the number of front stage messages.

gxf_c id	The GXF component ID.
gxf_c nam e	The name of the component.
gxf_c onte xt	The GXF context of the component.
gxf_e id	The GXF entity ID.
gxf_t ypen ame	The GXF type name of the condition.
id	The identifier of the component.
min_ size	The total number of messages over a set of input channels needed to permit execution.
nam e	The name of the condition.
recei ver	The receiver associated with the condition.

**spec**

## Methods

add_ arg (*args, **kwa rgs)	Overloaded function.
---	----------------------

<pre>gxf_initialize (self)</pre>	Initialize the component.
<pre>initialize (self)</pre>	Initialize the condition
<pre>setup (self, arg0)</pre>	Define the component specification.

```
__init__(self: holoscan.conditions._conditions.MessageAvailableCondition, fragment: holoscan.core._core.Fragment, min_size: int = 1, front_stage_max_size: int = 1, name: str = 'noname_message_available_condition') None
```

Condition that permits execution when an upstream message is available.

Satisfied when the associated receiver queue has at least a certain number of elements. The receiver is specified using the receiver parameter of the scheduling term. The minimum number of messages that permits the execution of the entity is specified by *min\_size*. An optional parameter for this scheduling term is *front\_stage\_max\_size*, the maximum front stage message count. If this parameter is set, the scheduling term will only allow execution if the number of messages in the queue does not exceed this count. It can be used for operators which do not consume all messages from the queue.

Parameters

**fragment**

The fragment the condition will be associated with

**min\_size**

The total number of messages over a set of input channels needed to permit execution.

**front\_stage\_max\_size**

Threshold for the number of front stage messages. Execution is only allowed if the number of front stage messages does not exceed this count.

**name**

The name of the condition.

`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* front\_stage\_max\_size

Threshold for the number of front stage messages. Execution is only allowed if the number of front stage messages does not exceed this count.

*property* `gxf_cid`

The GXF component ID.

*property* `gxf_cname`

The name of the component.

*property* `gxf_context`

The GXF context of the component.

*property* `gxf_eid`

The GXF entity ID.

`gxf_initialize(self: holoscan.gxf.\_gxf.GXFComponent)` None

Initialize the component.

*property* `gxf_typename`

The GXF type name of the condition.

Returns

str

The GXF type name of the condition

*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.conditions.\_conditions.MessageAvailableCondition)` None

Initialize the condition

This method is called only once when the condition is created for the first time, and uses a light-weight initialization.

*property* min\_size

The total number of messages over a set of input channels needed to permit execution.

*property* name

The name of the condition.

Returns

**name**

*property* receiver

The receiver associated with the condition.

`setup(self: holoscan.conditions.\_conditions.MessageAvailableCondition, arg0: holoscan.core.\_core.ComponentSpec)` None

Define the component specification.

Parameters

**spec**

Component specification associated with the condition.

*property* spec

`class holoscan.conditions.PeriodicCondition`

Bases: `holoscan.gxf._gxf.GXFCondition`

Condition class to support periodic execution of operators. The recess (pause) period indicates the minimum amount of time that must elapse before the `compute()` method can be executed again. The recess period can be specified as an integer value in nanoseconds.

For example: 1000 for 1 microsecond 1000000 for 1 millisecond, and 10000000000 for 1 second.

The recess (pause) period can also be specified as a `datetime.timedelta` object representing a duration. (see <https://docs.python.org/3/library/datetime.html#timedelta-objects>)

For example: `datetime.timedelta(minutes=1)`, `datetime.timedelta(seconds=1)`, `datetime.timedelta(milliseconds=1)` and `datetime.timedelta(microseconds=1)`. Supported argument names are: `weeks` | `days` | `hours` | `minutes` | `seconds` | `milliseconds` | `microseconds` This requires `import datetime`.

Parameters

### **fragment**

The fragment the condition will be associated with

### **recess\_period**

The recess (pause) period value used by the condition. If an integer is provided, the units are in nanoseconds.

### **name**

The name of the condition.

Attributes

<code>args</code>	The list of arguments associated with the component.
<code>description</code>	YAML formatted string describing the condition.

frag ment	Fragment that the condition belongs to.
gxf_ci d	The GXF component ID.
gxf_c nam e	The name of the component.
gxf_c onte xt	The GXF context of the component.
gxf_e id	The GXF entity ID.
gxf_t ypen ame	The GXF type name of the condition.
id	The identifier of the component.
nam e	The name of the condition.

## spec

### Methods

add_ arg (*args, **kwa rgs)	Overloaded function.
gxf_i nitiali ze (self)	Initialize the component.



<pre>initial ize (self)</pre>	Initialize the component.
<pre>last_r un_ti mest amp (self)</pre>	Gets the integer representing the last run time stamp.
<pre>reces s_per iod (*args, **kwa rgs)</pre>	Overloaded function.
<pre>reces s_per iod_n s (self)</pre>	Gets the recess (pause) period value in nanoseconds.
<pre>setu p (self, a rg0)</pre>	Define the component specification.

`__init__(*args, **kwargs)`

Overloaded function.

1. `__init__(self: holoscan.conditions._conditions.PeriodicCondition, fragment: holoscan.core._core.Fragment, recess_period: int, name: str = 'noname_periodic_condition') -> None`
2. `__init__(self: holoscan.conditions._conditions.PeriodicCondition, fragment: holoscan.core._core.Fragment, recess_period: datetime.timedelta, name: str = 'noname_periodic_condition') -> None`

Condition class to support periodic execution of operators. The recess (pause) period indicates the minimum amount of time that must elapse before the

`compute()` method can be executed again. The recess period can be specified as an integer value in nanoseconds.

For example: 1000 for 1 microsecond 1000000 for 1 millisecond, and 10000000000 for 1 second.

The recess (pause) period can also be specified as a `datetime.timedelta` object representing a duration. (see <https://docs.python.org/3/library/datetime.html#timedelta-objects>)

For example: `datetime.timedelta(minutes=1)`, `datetime.timedelta(seconds=1)`, `datetime.timedelta(milliseconds=1)` and `datetime.timedelta(microseconds=1)`. Supported argument names are: weeks | days | hours | minutes | seconds | milliseconds | microseconds This requires `import datetime`.

Parameters

### **fragment**

The fragment the condition will be associated with

### **recess\_period**

The recess (pause) period value used by the condition. If an integer is provided, the units are in nanoseconds.

### **name**

The name of the condition.

`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* gxf\_cid

The GXF component ID.

*property* gxf\_cname

The name of the component.

*property* gxf\_context

The GXF context of the component.

*property* gxf\_eid

The GXF entity ID.

`gxf_initialize(self: holoscan.gxf.\_gxf.GXFComponent)` None

Initialize the component.

*property* gxf\_typename

The GXF type name of the condition.

Returns

str

The GXF type name of the condition

*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.gxf.\_gxf.GXFCondition*) None

Initialize the component.

last\_run\_timestamp(*self: holoscan.conditions.\_conditions.PeriodicCondition*) int

Gets the integer representing the last run time stamp.

*property name*

The name of the condition.

Returns

**name**

recess\_period(*\*args, \*\*kwargs*)

Overloaded function.

1. recess\_period(*self: holoscan.conditions.\_conditions.PeriodicCondition, arg0: int*) -> None

Sets the recess (pause) period associated with the condition. The recess period can be specified as an integer value in nanoseconds or a *datetime.timedelta* object representing a duration.

2. `recess_period(self: holoscan.conditions._conditions.PeriodicCondition, arg0: datetime.timedelta) -> None`

Sets the recess (pause) period associated with the condition. The recess period can be specified as an integer value in nanoseconds or a *datetime.timedelta* object representing a duration.

`recess_period_ns(self: holoscan.conditions._conditions.PeriodicCondition) int`

Gets the recess (pause) period value in nanoseconds.

`setup(self: holoscan.conditions._conditions.PeriodicCondition, arg0: holoscan.core._core.ComponentSpec) None`

Define the component specification.

Parameters

**spec**

Component specification associated with the condition.

*property spec*

© Copyright 2022-2024, NVIDIA.. PDF Generated on 06/06/2024