# StandardExtension

# Table of contents

Most commonly used interfaces and components in Gxf Core.

- UUID: 8ec2d5d6-b5df-48bf-8dee-0252606fdd7e
- Version: 2.1.0
- Author: NVIDIA
- License: LICENSE

# Interfaces

## nvidia::gxf::Codelet

Interface for a component which can be executed to run custom code.

- Component ID: 5c6166fa-6eed-41e7-bbf0-bd48cd6e1014
- Base Type: nvidia::gxf::Component
- Defined in: gxf/std/codelet.hpp

## nvidia::gxf::Clock

Interface for clock components which provide time.

- Component ID: 779e61c2-ae70-441d-a26c-8ca64b39f8e7
- Base Type: nvidia::gxf::Component
- Defined in: gxf/std/clock.hpp

## nvidia::gxf::System

Component interface for systems which are run as part of the application run cycle.

- Component ID: d1febca1-80df-454e-a3f2-715f2b3c6e69
- Base Type: nvidia::gxf::Component

## nvidia::gxf::Queue

Interface for storing entities in a queue.

- Component ID: 792151bf-3138-4603-a912-5ca91828dea8
- Base Type: nvidia::gxf::Component
- Defined in: gxf/std/queue.hpp

## nvidia::gxf::Router

Interface for classes which are routing messages in and out of entities.

- Component ID: 8b317aad-f55c-4c07-8520-8f66db92a19e
- Defined in: gxf/std/router.hpp

## nvidia::gxf::Transmitter

Interface for publishing entities.

- Component ID: c30cc60f-0db2-409d-92b6-b2db92e02cce
- Base Type: nvidia::gxf::Queue
- Defined in: gxf/std/transmitter.hpp

## nvidia::gxf::Receiver

Interface for receiving entities.

- Component ID: a47d2f62-245f-40fc-90b7-5dc78ff2437e
- Base Type: nvidia::gxf::Queue
- Defined in: gxf/std/receiver.hpp

## nvidia::gxf::Scheduler

A simple poll-based single-threaded scheduler which executes codelets.

- Component ID: f0103b75-d2e1-4d70-9b13-3fe5b40209be

- Base Type: nvidia::gxf::System
- Defined in: nvidia/gxf/system.hpp

### nvidia::gxf::SchedulingTerm

Interface for terms used by a scheduler to determine if codelets in an entity are ready to step.

- Component ID: 184d8e4e-086c-475a-903a-69d723f95d19
- Base Type: nvidia::gxf::Component
- Defined in: gxf/std/scheduling_term.hpp

### nvidia::gxf::Allocator

Provides allocation and deallocation of memory.

- Component ID: 3cdd82d0-2326-4867-8de2-d565dbe28e03
- Base Type: nvidia::gxf::Component
- Defined in: nvidia/gxf/allocator.hpp

### nvidia::gxf::Monitor

Monitors entities during execution.

- Component ID: 9ccf9421-b35b-8c79-e1f0-97dc23bd38ea
- Base Type: nvidia::gxf::Component
- Defined in: nvidia/gxf/monitor.hpp

## Components

### nvidia::gxf::RealtimeClock

A real-time clock which runs based off a system steady clock.

- Component ID: 7b170b7b-cf1a-4f3f-997c-bfea25342381

- Base Type: nvidia::gxf::Clock

**Parameters**

**initial_time_offset**

The initial time offset used until time scale is changed manually.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_FLOAT64

**initial_time_scale**

The initial time scale used until time scale is changed manually.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_FLOAT64

**use_time_since_epoch**

If true, clock time is time since `epoch` + `initial_time_offset` at `initialize()`.Otherwise clock time is `initial_time_offset` at `initialize()`.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_BOOL

## nvidia::gxf::ManualClock

A manual clock which is instrumented manually.

- Component ID: 52fa1f97-eba8-472a-a8ca-4cff1a2c440f

- Base Type: nvidia::gxf::Clock

**Parameters**

**initial_timestamp**

The initial timestamp on the clock (in nanoseconds).

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_INT64

## nvidia::gxf::SystemGroup

A group of systems.

- Component ID: 3d23d470-0aed-41c6-ac92-685c1b5469a0
- Base Type: nvidia::gxf::System

## nvidia::gxf::MessageRouter

A router which sends transmitted messages to receivers.

- Component ID: 84fd5d56-fda6-4937-0b3c-c283252553d8
- Base Type: nvidia::gxf::Router

## nvidia::gxf::RouterGroup

A group of routers.

- Component ID: ca64ee14-2280-4099-9f10-d4b501e09117
- Base Type: nvidia::gxf::Router

## nvidia::gxf::DoubleBufferTransmitter

A transmitter which uses a double-buffered queue where messages are pushed to a backstage after they are published.

- Component ID: 0c3c0ec7-77f1-4389-aef1-6bae85bddc13

- Base Type: nvidia::gxf::Transmitter

**Parameters**

**capacity**

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_UINT64
- Default: 1

**policy**

0: pop, 1: reject, 2: fault.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_UINT64
- Default: 2

## nvidia::gxf::DoubleBufferReceiver

A receiver which uses a double-buffered queue where new messages are first pushed to a backstage.

- Component ID: ee45883d-bf84-4f99-8419-7c5e9deac6a5
- Base Type: nvidia::gxf::Receiver

**Parameters**

**capacity**

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_UINT64

- Default: 1

**policy**

0: pop, 1: reject, 2: fault

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_UINT64
- Default: 2

## nvidia::gxf::Connection

A component which establishes a connection between two other components.

- Component ID: cc71afae-5ede-47e9-b267-60a5c750a89a
- Base Type: nvidia::gxf::Component

**Parameters**

**source**

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Transmitter

**target**

- Flags: GXF_PARAMETER_FLAGS_NONE

- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Receiver

## nvidia::gxf::PeriodicSchedulingTerm

A component which specifies that an entity shall be executed periodically.

- Component ID: d392c98a-9b08-49b4-a422-d5fe6cd72e3e
- Base Type: nvidia::gxf::SchedulingTerm

**Parameters**

**recess_period**

The recess period indicates the minimum amount of time which has to pass before the entity is permitted to execute again. The period is specified as a string containing of a number and an (optional) unit. If no unit is given the value is assumed to be in nanoseconds. Supported units are: Hz, s, ms. Example: 10ms, 10000000, 0.2s, 50Hz.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_STRING

## nvidia::gxf::CountSchedulingTerm

A component which specifies that an entity shall be executed exactly a given number of times.

- Component ID: f89da2e4-fddf-4aa2-9a80-1119ba3fde05
- Base Type: nvidia::gxf::SchedulingTerm

**Parameters**

**count**

The total number of time this term will permit execution.

- Flags: GXF_PARAMETER_FLAGS_NONE

- Type: GXF_PARAMETER_TYPE_INT64

## nvidia::gxf::TargetTimeSchedulingTerm

A component where the next execution time of the entity needs to be specified after every tick.

- Component ID: e4aaf5c3-2b10-4c9a-c463-ebf6084149bf
- Base Type: nvidia::gxf::SchedulingTerm

**Parameters**

**clock**

The clock used to define target time.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Clock

## nvidia::gxf::DownstreamReceptiveSchedulingTerm

A component which specifies that an entity shall be executed if receivers for a certain transmitter can accept new messages.

- Component ID: 9de75119-8d0f-4819-9a71-2aeaefd23f71
- Base Type: nvidia::gxf::SchedulingTerm

**Parameters**

**min_size**

The term permits execution if the receiver connected to the transmitter has at least the specified number of free slots in its back buffer.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_UINT64

**transmitter**

The term permits execution if this transmitter can publish a message, i.e. if the receiver which is connected to this transmitter can receive messages.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Transmitter

## nvidia::gxf::MessageAvailableSchedulingTerm

A scheduling term which specifies that an entity can be executed when the total number of messages over a set of input channels is at least a given number of messages.

- Component ID: fe799e65-f78b-48eb-beb6-e73083a12d5b
- Base Type: nvidia::gxf::SchedulingTerm

**Parameters**

**front_stage_max_size**

If set the scheduling term will only allow execution if the number of messages in the front stage does not exceed this count. It can for example be used in combination with codelets which do not clear the front stage in every tick.

- Flags: GXF_PARAMETER_FLAGS_OPTIONAL
- Type: GXF_PARAMETER_TYPE_UINT64

**min_size**

The scheduling term permits execution if the given receiver has at least the given number of messages available.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_UINT64

**receiver**

The scheduling term permits execution if this channel has at least a given number of messages available.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Receiver

## nvidia::gxf::MultiMessageAvailableSchedulingTerm

A component which specifies that an entity shall be executed when a queue has at least a certain number of elements.

- Component ID: f15dbeaa-afd6-47a6-9ffc-7afd7e1b4c52
- Base Type: nvidia::gxf::SchedulingTerm

**Parameters**

**min_size**

The scheduling term permits execution if all given receivers together have at least the given number of messages available.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_UINT64

**receivers**

The scheduling term permits execution if the given channels have at least a given number of messages available.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Receiver

## nvidia::gxf::ExpiringMessageAvailableSchedulingTerm

A component which tries to wait for specified number of messages in queue for at most specified time.

- Component ID: eb22280c-76ff-11eb-b341-cf6b417c95c9
- Base Type: nvidia::gxf::SchedulingTerm

**Parameters**

**clock**

Clock to get time from.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Clock

**max_batch_size**

The maximum number of messages to be batched together.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_INT64

**max_delay_ns**

The maximum delay from first message to wait before submitting workload anyway.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_INT64

**receiver**

Receiver to watch on.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Receiver

## nvidia::gxf::BooleanSchedulingTerm

A component which acts as a boolean AND term that can be used to control the execution of the entity.

- Component ID: e07a0dc4-3908-4df8-8134-7ce38e60fbef
- Base Type: nvidia::gxf::SchedulingTerm

## nvidia::gxf::AsynchronousSchedulingTerm

A component which is used to inform of that an entity is dependent upon an async event for its execution.

- Component ID: 56be1662-ff63-4179-9200-3fcd8dc38673
- Base Type: nvidia::gxf::SchedulingTerm

## nvidia::gxf::GreedyScheduler

A simple poll-based single-threaded scheduler which executes codelets.

- Component ID: 869d30ca-a443-4619-b988-7a52e657f39b
- Base Type: nvidia::gxf::Scheduler

**Parameters**

**clock**

The clock used by the scheduler to define flow of time. Typical choices are a `RealtimeClock` or a `ManualClock`.

- Flags: GXF_PARAMETER_FLAGS_OPTIONAL
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Clock

**max_duration_ms**

The maximum duration for which the scheduler will execute (in ms). If not specified the scheduler will run until all work is done. If periodic terms are present this means the application will run indefinitely.

- Flags: GXF_PARAMETER_FLAGS_OPTIONAL
- Type: GXF_PARAMETER_TYPE_INT64

**realtime**

This parameter is deprecated. Assign a clock directly.

- Flags: GXF_PARAMETER_FLAGS_OPTIONAL
- Type: GXF_PARAMETER_TYPE_BOOL

**stop_on_deadlock**

If enabled the scheduler will stop when all entities are in a waiting state, but no periodic entity exists to break the dead end. Should be disabled when scheduling conditions can be changed by external actors, for example by clearing queues manually.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_BOOL

## nvidia::gxf::MultiThreadScheduler

A multi thread scheduler that executes codelets for maximum throughput.

- Component ID: de5e0646-7fa5-11eb-a5c4-330ebfa81bbf
- Base Type: nvidia::gxf::Scheduler

**Parameters**

**check_recession_perios_ms**

The maximum duration for which the scheduler would wait (in ms) when an entity is not ready to run yet.

- Flags: GXF_PARAMETER_FLAGS_NONE

- Type: GXF_PARAMETER_TYPE_INT64

**clock**

The clock used by the scheduler to define flow of time. Typical choices are a `RealtimeClock` or a `ManualClock`.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Clock

**max_duration_ms**

The maximum duration for which the scheduler will execute (in ms). If not specified the scheduler will run until all work is done. If periodic terms are present this means the application will run indefinitely.

- Flags: GXF_PARAMETER_FLAGS_OPTIONAL
- Type: GXF_PARAMETER_TYPE_INT64

**stop_on_deadlock**

If enabled the scheduler will stop when all entities are in a waiting state, but no periodic entity exists to break the dead end. Should be disabled when scheduling conditions can be changed by external actors, for example by clearing queues manually.

- Flags: GXF_PARAMETER_FLAGS_NONE

- Type: GXF_PARAMETER_TYPE_BOOL

**worker_thread_number**

Number of threads.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_INT64
- Default: 1

## nvidia::gxf::BlockMemoryPool

A memory pools which provides a maximum number of equally sized blocks of memory.

- Component ID: 92b627a3-5dd3-4c3c-976c-4700e8a3b96a
- Base Type: nvidia::gxf::Allocator

**Parameters**

**block_size**

The size of one block of memory in byte. Allocation requests can only be fulfilled if they fit into one block. If less memory is requested still a full block is issued.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_UINT64

**do_not_use_cuda_malloc_host**

If enabled operator new will be used to allocate host memory instead of `cudaMallocHost`.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_BOOL
- Default: True

**num_blocks**

The total number of blocks which are allocated by the pool. If more blocks are requested allocation requests will fail.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_UINT64

**storage_type**

The memory storage type used by this allocator. Can be kHost (0) or kDevice (1) or kSystem (2).

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_INT32
- Default: 0

## nvidia::gxf::UnboundedAllocator

Allocator that uses dynamic memory allocation without an upper bound.

- Component ID: c3951b16-a01c-539f-d87e-1dc18d911ea0

- Base Type: nvidia::gxf::Allocator

**Parameters**

**do_not_use_cuda_malloc_host**

If enabled operator new will be used to allocate host memory instead of `cudaMallocHost`.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_BOOL
- Default: True

## nvidia::gxf::Tensor

A component which holds a single tensor.

- Component ID: 377501d6-9abf-447c-a617-0114d4f33ab8
- Defined in: gxf/std/tensor.hpp

## nvidia::gxf::Timestamp

Holds message publishing and acquisition related timing information.

- Component ID: d1095b10-5c90-4bbc-bc89-601134cb4e03
- Defined in: gxf/std/timestamp.hpp

## nvidia::gxf::Metric

Collects, aggregates, and evaluates metric data.

- Component ID: f7cef803-5beb-46f1-186a-05d3919842ac
- Base Type: nvidia::gxf::Component

**Parameters**

**aggregation_policy**

Aggregation policy used to aggregate individual metric samples. Choices:{mean, min, max}.

- Flags: GXF_PARAMETER_FLAGS_OPTIONAL
- Type: GXF_PARAMETER_TYPE_STRING

**lower_threshold**

Lower threshold of the metric's expected range.

- Flags: GXF_PARAMETER_FLAGS_OPTIONAL
- Type: GXF_PARAMETER_TYPE_FLOAT64

**upper_threshold**

Upper threshold of the metric's expected range.

- Flags: GXF_PARAMETER_FLAGS_OPTIONAL
- Type: GXF_PARAMETER_TYPE_FLOAT64

## nvidia::gxf::JobStatistics

Collects runtime statistics.

- Component ID: 2093b91a-7c82-11eb-a92b-3f1304ecc959
- Base Type: nvidia::gxf::Component

**Parameters**

**clock**

The clock component instance to retrieve time from.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Clock

**codelet_statistics**

If set to true, JobStatistics component will collect performance statistics related to codelets.

- Flags: GXF_PARAMETER_FLAGS_OPTIONAL
- Type: GXF_PARAMETER_TYPE_BOOL

**json_file_path**

If provided, all the collected performance statistics data will be dumped into a json file.

- Flags: GXF_PARAMETER_FLAGS_OPTIONAL
- Type: GXF_PARAMETER_TYPE_STRING

## nvidia::gxf::Broadcast

Messages arrived on the input channel are distributed to all transmitters.

- Component ID: 3daadb31-0bca-47e5-9924-342b9984a014
- Base Type: nvidia::gxf::Codelet

**Parameters**

**mode**

The broadcast mode. Can be Broadcast or RoundRobin.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_CUSTOM

**source**

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Receiver

## nvidia::gxf::Gather

All messages arriving on any input channel are published on the single output channel.

- Component ID: 85f64c84-8236-4035-9b9a-3843a6a2026f
- Base Type: nvidia::gxf::Codelet

**Parameters**

**sink**

The output channel for gathered messages.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Transmitter

**tick_source_limit**

Maximum number of messages to take from each source in one tick. 0 means no limit.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_INT64

## nvidia::gxf::TensorCopier

Copies tensor either from host to device or from device to host.

- Component ID: c07680f4-75b3-189b-8886-4b5e448e7bb6
- Base Type: nvidia::gxf::Codelet

**Parameters**

**allocator**

Memory allocator for tensor data

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Allocator

**mode**

Configuration to select what tensors to copy:

1. kCopyToDevice (0) - copies to device memory, ignores device allocation
2. kCopyToHost (1) - copies to pinned host memory, ignores host allocation

3. kCopyToSystem (2) - copies to system memory, ignores system allocation.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_INT32

**receiver**

Receiver for incoming entities.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Receiver

**transmitter**

Transmitter for outgoing entities.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Transmitter

## nvidia::gxf::TimedThrottler

Publishes the received entity respecting the timestamp within the entity.

- Component ID: ccf7729c-f62c-4250-5cf7-f4f3ec80454b
- Base Type: nvidia::gxf::Codelet

**Parameters**

**execution_clock**

Clock on which the codelet is executed by the scheduler.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Clock

**receiver**

Channel to receive messages that need to be synchronized.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Receiver

**scheduling_term**

Scheduling term for executing the codelet.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::TargetTimeSchedulingTerm

**throttling_clock**

Clock which the received entity timestamps are based on.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Clock

**transmitter**

Transmitter channel publishing messages at appropriate timesteps.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Transmitter

## nvidia::gxf::Vault

Safely stores received entities for further processing.

- Component ID: 1108cb8d-85e4-4303-ba02-d27406ee9e65
- Base Type: nvidia::gxf::Codelet

**Parameters**

**drop_waiting**

If too many messages are waiting the oldest ones are dropped.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_BOOL

#### max_waiting_count

The maximum number of waiting messages. If exceeded the codelet will stop pulling messages out of the input queue.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_UINT64

#### source

Receiver from which messages are taken and transferred to the vault.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Receiver

### nvidia::gxf::Subgraph

Helper component to import a subgraph.

- Component ID: 576eedd7-7c3f-4d2f-8c38-8baa79a3d231
- Base Type: nvidia::gxf::Component

#### Parameters

#### location

`Yaml` source of the subgraph.

- Flags: GXF_PARAMETER_FLAGS_NONE

- Type: GXF_PARAMETER_TYPE_STRING

## nvidia::gxf::EndOfStream

A component which represents end-of-stream notification.

- Component ID: 8c42f7bf-7041-4626-9792-9eb20ce33cce
- Defined in: gxf/std/eos.hpp

## nvidia::gxf::Synchronization

Component to synchronize messages from multiple receivers based on the `acq_time`.

- Component ID: f1cb80d6-e5ec-4dba-9f9e-b06b0def4443
- Base Type: nvidia::gxf::Codelet

**Parameters**

**inputs**

All the inputs for synchronization. Number of inputs must match that of the outputs.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE
- Handle Type: nvidia::gxf::Receiver

**outputs**

All the outputs for synchronization. Number of outputs must match that of the inputs.

- Flags: GXF_PARAMETER_FLAGS_NONE
- Type: GXF_PARAMETER_TYPE_HANDLE

- Handle Type: nvidia::gxf::Transmitter

### signed char

- Component ID: 83905c6a-ca34-4f40-b474-cf2cde8274de

### unsigned char

- Component ID: d4299e15-0006-d0bf-8cbd-9b743575e155

### short int

- Component ID: 9e1dde79-3550-307d-e81a-b864890b3685

### short unsigned int

- Component ID: 958cbdef-b505-bcc7-8a43-dc4b23f8cead

### int

- Component ID: b557ec7f-49a5-08f7-a35e-086e9d1ea767

### unsigned int

- Component ID: d5506b68-5c86-fedb-a2a2-a7bae38ff3ef

### long int

- Component ID: c611627b-6393-365f-d234-1f26bfa8d28f

### long unsigned int

- Component ID: c4385f5b-6e25-01d9-d7b5-6e7cadc704e8

### float

- Component ID: a81bf295-421f-49ef-f24a-f59e9ea0d5d6

### double

- Component ID: d57cee59-686f-e26d-95be-659c126b02ea

## bool

- Component ID: c02f9e93-d01b-1d29-f523-78d2a9195128

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