



Program Listing for File serialization_buffer.hpp

[Return to documentation for file \(](#)

`include/holoscan/core/resources/gxf/serialization_buffer.hpp`)

```
/* * SPDX-FileCopyrightText: Copyright (c) 2023-2024 NVIDIA CORPORATION &
AFFILIATES. All rights reserved. * SPDX-License-Identifier: Apache-2.0 * * Licensed
under the Apache License, Version 2.0 (the "License"); * you may not use this file
except in compliance with the License. * You may obtain a copy of the License at * *
http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law
or agreed to in writing, software * distributed under the License is distributed on an
"AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied. * See the License for the specific language governing
permissions and * limitations under the License. */ #ifndef
HOLOSCAN_CORE_RESOURCES_GXF_SERIALIZATION_BUFFER_HPP #define
HOLOSCAN_CORE_RESOURCES_GXF_SERIALIZATION_BUFFER_HPP #include
<cstdint> #include <memory> #include <string> #include
<gxf/serialization/serialization_buffer.hpp> #include "../gxf/gxf_resource.hpp"
#include "../unbounded_allocator.hpp" namespace holoscan { constexpr size_t
kDefaultSerializationBufferSize = 1 << 12; // 4 kB class SerializationBuffer : public
gxf::GXFResource { public:
HOLOSCAN_RESOURCE_FORWARD_ARGS_SUPER(SerializationBuffer, GXFResource)
SerializationBuffer() = default; SerializationBuffer(const std::string& name,
nvidia::gxf::SerializationBuffer* component); const char* gxf_typename() const
override { return "nvidia::gxf::SerializationBuffer"; } void setup(ComponentSpec&
spec) override; void initialize() override; nvidia::gxf::SerializationBuffer* get() const;
private: Parameter<std::shared_ptr<holoscan::Allocator>> allocator_;
Parameter<size_t> buffer_size_; }; } // namespace holoscan #endif/*
HOLOSCAN_CORE_RESOURCES_GXF_SERIALIZATION_BUFFER_HPP */
```

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