Program Listing for File allocator.hpp

Return to documentation for file (include/holoscan/core/resources/gxf/allocator.hpp)

/* * SPDX-FileCopyrightText: Copyright (c) 2022-2024 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_ALLOCATOR_HPP #define HOLOSCAN_CORE_RESOURCES_GXF_ALLOCATOR_HPP #include <string> #include <gxf/std/allocator.hpp> #include "../../gxf/gxf_resource.hpp" namespace holoscan { enum struct MemoryStorageType { kHost = 0, kDevice = 1, kSystem = 2 }; class Allocator : public gxf::GXFResource { public: HOLOSCAN_RESOURCE_FORWARD_ARGS_SUPER(Allocator, GXFResource) Allocator() = default; Allocator(const std::string& name, nvidia::gxf::Allocator* component); const char* gxf_typename() const override { return "nvidia::gxf::Allocator"; } virtual bool is_available(uint64_t size); *// TODO(gbae): Introduce expected<> type* virtual nvidia::byte* allocate(uint64_t size, MemoryStorageType type); virtual void free(nvidia::byte* pointer); *// Get the block size of this allocator, defaults to 1 for byte-based allocators* uint64_t block_size(); nvidia::gxf::Allocator* get() const; }; } *// namespace holoscan* #endif/* HOLOSCAN_CORE_RESOURCES_GXF_ALLOCATOR_HPP */