Program Listing for File expected.hpp

Return to documentation for file (include/holoscan/core/expected.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_EXPECTED_HPP #define HOLOSCAN_CORE_EXPECTED_HPP #include <tl/expected.hpp> #include <utility> namespace holoscan { template <class T, class E> using expected = tl::expected<T, E>; template <class E> using unexpected = tl::unexpected<E>; template <class E> using bad_expected_access = tl::bad_expected_access<E>; using unexpect_t = tl::unexpect_t; static constexpr unexpect_t unexpect{}; *// codespell-ignore* template <class E> static inline constexpr unexpected<E> make_unexpected(E&& e) { return unexpected<E>{std::forward<E>(e)}; } *// Extracts the error code as an unexpected.* template <class T, class E> unexpected<E> forward_error(const expected<T, E>& expected) { return unexpected<E>{expected.error()}; } *// Extracts the error code as an unexpected.* template <class T, class E> unexpected<E> forward_error(expected<T, E>&& expected) { return make_unexpected(expected.error()); } } *// namespace holoscan* #endif/* HOLOSCAN_CORE_EXPECTED_HPP */