Program Listing for File nvml_wrapper.h

Return to documentation for file (`include/holoscan/core/system/nvml_wrapper.h`)

```
/* * SPDX-FileCopyrightText: Copyright (c) 2023 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_SYSTEM_NVML_WRAPPER_H #define
HOLOSCAN_CORE_SYSTEM_NVML_WRAPPER_H namespace holoscan::nvml { // The
full list of methods that NVML provides is available here: //
https://docs.nvidia.com/deploy/nvml-api/group__nvmlDeviceQueries.html // We wrap
only the methods that we need for the GPUInfo class. typedef struct nvmlDevice_st*
nvmlDevice_t; typedef struct nvmlMemory_st { unsigned long long total; unsigned
long long free; unsigned long long used; } nvmlMemory_t; #define
NVML_DEVICE_NAME_BUFFER_SIZE 64 #define
NVML_DEVICE_PCI_BUS_ID_BUFFER_SIZE 32 #define
NVML_DEVICE_PCI_BUS_ID_BUFFER_V2_SIZE 16 #define
NVML_DEVICE_SERIAL_BUFFER_SIZE 30 #define NVML_DEVICE_UUID_BUFFER_SIZE
80 typedef struct nvmlPciInfo_st { char
busIdLegacy[NVML_DEVICE_PCI_BUS_ID_BUFFER_V2_SIZE]; unsigned int domain;
unsigned int bus; unsigned int device; unsigned int pciDeviceId; // Added in NVML
2.285 API unsigned int pciSubSystemId; char
busId[NVML_DEVICE_PCI_BUS_ID_BUFFER_SIZE]; } nvmlPciInfo_t; typedef struct
nvmlUtilization_st { unsigned int gpu; unsigned int memory; } nvmlUtilization_t;
enum nvmlTemperatureSensors_t { NVML_TEMPERATURE_GPU = 0,
NVML_TEMPERATURE_COUNT }; typedef int nvmlReturn_t; // const char*
nvmlErrorString ( nvmlReturn_t result ) typedef const char* (*nvmlErrorString_t)
(nvmlReturn_t); // nvmlReturn_t nvmlInit_v2 ( void ) typedef nvmlReturn_t (*nvmlInit_t)
(); // nvmlReturn_t nvmlDeviceGetCount_v2 ( unsigned int* deviceCount ) typedef
nvmlReturn_t (*nvmlDeviceGetCount_t)(unsigned int*); // nvmlReturn_t
nvmlDeviceGetHandleByIndex_v2 ( unsigned int index, nvmlDevice_t* device ) typedef
nvmlReturn_t (*nvmlDeviceGetHandleByIndex_t)(unsigned int, nvmlDevice_t*); //
nvmlReturn_t nvmlDeviceGetHandleByPciBusId_v2 ( const char* pciBusId, nvmlDevice_t*
device ) typedef nvmlReturn_t (*nvmlDeviceGetHandleByPciBusId_t)(const char*,
nvmlDevice_t*); // nvmlReturn_t nvmlDeviceGetHandleBySerial ( const char* serial,
nvmlDevice_t* device ) typedef nvmlReturn_t (*nvmlDeviceGetHandleBySerial_t)
(const char*, nvmlDevice_t*); // nvmlReturn_t nvmlDeviceGetHandleByUUID ( const
char* uuid, nvmlDevice_t* device ) typedef nvmlReturn_t
(*nvmlDeviceGetHandleByUUID_t)(const char*, nvmlDevice_t*); // nvmlReturn_t
nvmlDeviceGetName ( nvmlDevice_t device, char* name, unsigned int length ) typedef
nvmlReturn_t (*nvmlDeviceGetName_t)(nvmlDevice_t, char*, unsigned int); //
nvmlReturn_t nvmlDeviceGetIndex ( nvmlDevice_t device, unsigned int* index ) typedef
nvmlReturn_t (*nvmlDeviceGetIndex_t)(nvmlDevice_t, unsigned int*); // nvmlReturn_t
nvmlDeviceGetPciInfo_v3 ( nvmlDevice_t device, nvmlPciInfo_t* pci ) typedef
nvmlReturn_t (*nvmlDeviceGetPciInfo_t)(nvmlDevice_t, nvmlPciInfo_t*); //
nvmlReturn_t nvmlDeviceGetSerial ( nvmlDevice_t device, char* serial, unsigned int
length ) typedef nvmlReturn_t (*nvmlDeviceGetSerial_t)(nvmlDevice_t, char*,
unsigned int); // nvmlReturn_t nvmlDeviceGetUUID ( nvmlDevice_t device, char* uuid,
unsigned int length ) typedef nvmlReturn_t (*nvmlDeviceGetUUID_t)(nvmlDevice_t,
char*, unsigned int); // nvmlReturn_t nvmlDeviceGetMemoryInfo ( nvmlDevice_t device,
nvmlMemory_t* memory ) typedef nvmlReturn_t (*nvmlDeviceGetMemoryInfo_t)
(nvmlDevice_t, nvmlMemory_t*); // nvmlReturn_t nvmlDeviceGetUtilizationRates (
nvmlDevice_t device, nvmlUtilization_t* utilization // ) typedef nvmlReturn_t
(*nvmlDeviceGetUtilizationRates_t)(nvmlDevice_t, nvmlUtilization_t*); // nvmlReturn_t
nvmlDeviceGetPowerManagementLimit ( nvmlDevice_t device, unsigned int* limit )
typedef nvmlReturn_t (*nvmlDeviceGetPowerManagementLimit_t)(nvmlDevice_t,
unsigned int*); // nvmlReturn_t nvmlDeviceGetPowerUsage ( nvmlDevice_t device,
unsigned int* power ) typedef nvmlReturn_t (*nvmlDeviceGetPowerUsage_t)
(nvmlDevice_t, unsigned int*); // nvmlReturn_t nvmlDeviceGetTemperature (
nvmlDevice_t device, nvmlTemperatureSensors_t sensorType, // unsigned int* temp )
typedef nvmlReturn_t (*nvmlDeviceGetTemperature_t)(nvmlDevice_t, unsigned int,
unsigned int*); // nvmlReturn_t nvmlShutdown ( void ) typedef nvmlReturn_t
(*nvmlShutdown_t)(); } // namespace holoscan::nvml #endif/*
HOLOSCAN_CORE_SYSTEM_NVML_WRAPPER_H */
```