



MT GPU Management Library (MTML)

API Reference

Version: v 1.1.0

2022-09-06

1 Main Page	1
1.1 Basic concepts	1
1.1.1 Supported platforms	1
1.1.2 Opaque data types	1
1.1.3 Initialization and freeing	2
1.1.4 Virtualization	2
1.1.5 Codec session	2
2 Module Index	3
2.1 Modules	3
3 Class Index	4
3.1 Class List	4
4 File Index	5
4.1 File List	5
5 Module Documentation	6
5.1 Constant Definitions	6
5.1.1 Detailed Description	7
5.1.2 Macro Definition Documentation	7
5.1.2.1 MTML_DEVICE_NAME_BUFFER_SIZE	7
5.1.2.2 MTML_DEVICE_PATH_BUFFER_SIZE	7
5.1.2.3 MTML_DEVICE_PCI_BUS_ID_FMT	7
5.1.2.4 MTML_DEVICE_PCI_SBDF_BUFFER_SIZE	7
5.1.2.5 MTML_DEVICE_UUID_BUFFER_SIZE	7
5.1.2.6 MTML_DEVICE_VBIOS_VERSION_BUFFER_SIZE	7
5.1.2.7 MTML_DRIVER_VERSION_BUFFER_SIZE	8
5.1.2.8 MTML_LIBRARY_VERSION_BUFFER_SIZE	8
5.1.2.9 MTML_VIRT_TYPE_API_BUFFER_SIZE	8
5.1.2.10 MTML_VIRT_TYPE_CLASS_BUFFER_SIZE	8
5.1.2.11 MTML_VIRT_TYPE_ID_BUFFER_SIZE	8
5.1.2.12 MTML_VIRT_TYPE_NAME_BUFFER_SIZE	8
5.1.3 Enumeration Type Documentation	8
5.1.3.1 MtmlBrandType	8
5.1.3.2 MtmlCodecType	9
5.1.3.3 MtmlReturn	9
5.1.3.4 MtmlVirtRole	9
5.2 Functional Structure Definitions	10
5.2.1 Detailed Description	10
5.3 Opaque Data Structure Definitions	10
5.3.1 Detailed Description	10
5.3.2 Typedef Documentation	10
5.3.2.1 MtmlDevice	10

5.3.2.2 MtmlGpu	10
5.3.2.3 MtmlLibrary	11
5.3.2.4 MtmlMemory	11
5.3.2.5 MtmlSystem	11
5.3.2.6 MtmlVpu	11
5.4 Library Functions	11
5.4.1 Detailed Description	11
5.4.2 Function Documentation	11
5.4.2.1 mtmLibraryCountDevice()	11
5.4.2.2 mtmLibraryFreeDevice()	12
5.4.2.3 mtmLibraryFreeSystem()	12
5.4.2.4 mtmLibraryGetVersion()	13
5.4.2.5 mtmLibraryInit()	13
5.4.2.6 mtmLibraryInitDeviceByIndex()	14
5.4.2.7 mtmLibraryInitDeviceByUuid()	15
5.4.2.8 mtmLibraryInitSystem()	16
5.4.2.9 mtmLibraryShutDown()	16
5.5 System Functions	17
5.5.1 Detailed Description	17
5.5.2 Function Documentation	17
5.5.2.1 mtmSystemGetDriverVersion()	17
5.6 Device Functions	17
5.6.1 Detailed Description	18
5.6.2 Function Documentation	18
5.6.2.1 mtmDeviceFreeGpu()	18
5.6.2.2 mtmDeviceFreeMemory()	18
5.6.2.3 mtmDeviceFreeVpu()	19
5.6.2.4 mtmDeviceGetBrand()	19
5.6.2.5 mtmDeviceGetGpuPath()	20
5.6.2.6 mtmDeviceGetIndex()	20
5.6.2.7 mtmDeviceGetName()	21
5.6.2.8 mtmDeviceGetPciInfo()	21
5.6.2.9 mtmDeviceGetPowerUsage()	22
5.6.2.10 mtmDeviceGetPrimaryPath()	22
5.6.2.11 mtmDeviceGetProperty()	23
5.6.2.12 mtmDeviceGetRenderPath()	23
5.6.2.13 mtmDeviceGetUUID()	24
5.6.2.14 mtmDeviceGetVbiosVersion()	24
5.6.2.15 mtmDeviceInitGpu()	25
5.6.2.16 mtmDeviceInitMemory()	26
5.6.2.17 mtmDeviceInitVpu()	26
5.7 Device Virtualization Functions	27

5.7.1 Detailed Description	27
5.7.2 Function Documentation	27
5.7.2.1 mtmIDeviceCountActiveVirtDevices()	27
5.7.2.2 mtmIDeviceCountAvailVirtDevices()	28
5.7.2.3 mtmIDeviceCountAvailVirtTypes()	28
5.7.2.4 mtmIDeviceCountSupportedVirtTypes()	29
5.7.2.5 mtmIDeviceFreeVirtDevice()	30
5.7.2.6 mtmIDeviceGetActiveVirtDeviceUuids()	30
5.7.2.7 mtmIDeviceGetAvailVirtTypes()	31
5.7.2.8 mtmIDeviceGetPhyDeviceUuid()	31
5.7.2.9 mtmIDeviceGetSupportedVirtTypes()	32
5.7.2.10 mtmIDeviceGetVirtType()	33
5.7.2.11 mtmIDeviceInitVirtDevice()	33
5.8 GPU Functions	34
5.8.1 Detailed Description	34
5.8.2 Function Documentation	34
5.8.2.1 mtmIGpuGetClock()	34
5.8.2.2 mtmIGpuGetMaxClock()	35
5.8.2.3 mtmIGpuGetTemperature()	35
5.8.2.4 mtmIGpuGetUtilization()	36
5.9 Memory Functions	36
5.9.1 Detailed Description	36
5.9.2 Function Documentation	36
5.9.2.1 mtmIMemoryGetClock()	36
5.9.2.2 mtmIMemoryGetMaxClock()	37
5.9.2.3 mtmIMemoryGetTotal()	37
5.9.2.4 mtmIMemoryGetUsed()	38
5.9.2.5 mtmIMemoryGetUtilization()	38
5.10 VPU Functions	39
5.10.1 Detailed Description	39
5.10.2 Function Documentation	39
5.10.2.1 mtmIVpuGetClock()	39
5.10.2.2 mtmIVpuGetCodecCapacity()	40
5.10.2.3 mtmIVpuGetDecoderSessionMetrics()	40
5.10.2.4 mtmIVpuGetDecoderSessionStates()	41
5.10.2.5 mtmIVpuGetEncoderSessionMetrics()	42
5.10.2.6 mtmIVpuGetEncoderSessionStates()	42
5.10.2.7 mtmIVpuGetMaxClock()	43
5.10.2.8 mtmIVpuGetUtilization()	43
6 Class Documentation	45
6.1 MtmIPciInfo Struct Reference	45

6.1.1 Detailed Description	46
6.2 MtmlCodecUtil Struct Reference	46
6.2.1 Detailed Description	46
6.3 MtmlCodecSessionMetrics Struct Reference	46
6.3.1 Detailed Description	47
6.4 MtmlVirtType Struct Reference	47
6.4.1 Detailed Description	47
6.5 MtmlDeviceProperty Struct Reference	47
6.5.1 Detailed Description	48
6.5.2 Member Data Documentation	48
6.5.2.1 virtCap	48
6.5.2.2 virtRole	48
7 File Documentation	49
7.1 mtml.h	49
Index	54

Chapter 1

Main Page

1.1 Basic concepts

1.1.1 Supported platforms

CPU architecture	OS
x86_64	Ubuntu 20.04, UOS 20, and Kylin V10
AArch64	Kylin V10 SP1

1.1.2 Opaque data types

The MTML library works with several core opaque data types. Each data type represents a specific functional entity that a user can interact with. Most functions defined below take one or more opaque objects as its input parameters. Therefore, it is important for a user to have a good understanding of them.

- **MtmlLibrary** represents the MTML library itself and plays as the entry point to access and create other opaque objects.
- **MtmlSystem** represents the system environment in which the library is running.
- **MtmlDevice** represents the Moore Threads device (including virtual devices) that is installed in the system.
- **MtmlGpu** represents the graphic unit of a Moore Threads device, which is responsible for the 3D and compute workloads.
- **MtmlMemory** represents the memory units that reside on a Moore Threads device.
- **MtmlVpu** represents the video codec unit of a Moore Threads device which handles the video encoding and decoding task.

The relationship among the above opaque data types is hierarchical, which means some type 'contains' other types. The hierarchy of opaque data types mentioned above can be summarized as follows.

```
Library
|--- System
|--- Device
|       |--- GPU
|       |--- Memory
|       |--- VPU
```

1.1.3 Initialization and freeing

It is important to properly initialize an opaque object before using it as an argument to invoke other functions. The hierarchical relationships among opaque data types determine the initialization order of them. For example, to initialize an `MtmlGpu` struct, a user should firstly initialize a `MtmlLibrary` struct, then initialize the `MtmlDevice` struct from the `MtmlLibrary`, and finally initialize the `MtmlGpu` struct from the `MtmlDevice` struct.

Initializing an opaque object is completed by invoking an `mtml***Init***()` function. When an opaque object is not needed any longer, it's the user's responsibility to instruct the MTML library to release the underlying resources of the opaque object by calling the corresponding `mtml***Free***()` function. Due to the hierarchical relationships among opaque data types, it is important to release the opaque data pointers in reversed order as regards to their initialization order.

1.1.4 Virtualization

It is recommended to build some fundamental concepts before using the device virtualization management feature provided by the MTML library. There are two major functionalities with regards to the virtualization feature: virtualization type management and virtual device management. More specifically, several concepts shall be introduced:

- **Virtualizability** - This is a concept used to describe whether a physical device supports virtualization. Therefore, a virtualizable device is defined as a physical device from which virtual resources can be created.
- **Virtualization Type** - A virtualization type is a specification template that confines virtual resource (for example, device memory) allocation. Typically, a virtualizable device supports several virtualization types, therefore virtual devices (see below) with corresponding specifications can be created.
- **Virtual Device** - A virtual device is created from a virtualizable device according to a specified virtualization type. It is key to remember that a virtual device is a host-only concept. That means you can initialize an `MtmlDevice` opaque object that represents a virtual device only if the MTML library is running in a virtualization-enabled host environment.

1.1.5 Codec session

A codec session is an independent encoding/decoding process of a video stream. The concurrent codec sessions number that can be supported by the video codec unit of a Moore Threads device is called **codec capacity**. The **decodeCapacity** and **encodeCapacity** represent the maximum limit of concurrent decoder sessions and encoder sessions, respectively.

Each codec session is identifiable from a unique integral value, called **session ID**. For decoder sessions, the session ID ranges from **0** ~ (`decodeCapacity`-1), while it ranges from **0** ~ (`encodeCapacity`-1) for encoder sessions.

Another important aspect of a codec session is its **state**. The definition of the state of a codec session is as follows:

- **Inactive** (Idle): There is no video workload assigned to this session.
- **Active**: A video encoding/decoding workload is assigned to this session. At any given time point, the state of a codec session can be queried. If a codec session is in an active state, its metrics information then can be queried.

Refer to the `videoMonApp()` sample in `sample/basic/basic_sample.cpp` for a workable sample code.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Constant Definitions	6
Functional Structure Definitions	10
Opaque Data Structure Definitions	10
Library Functions	11
System Functions	17
Device Functions	17
Device Virtualization Functions	27
GPU Functions	34
Memory Functions	36
VPU Functions	39

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

MtmlPciInfo	45
MtmlCodecUtil	46
MtmlCodecSessionMetrics	46
MtmlVirtType	47
MtmlDeviceProperty	47

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

mtml.h	49
----------------------------------	----

Chapter 5

Module Documentation

5.1 Constant Definitions

Macros

- #define [MTML_LIBRARY_VERSION_BUFFER_SIZE](#) 32
- #define [MTML_DRIVER_VERSION_BUFFER_SIZE](#) 80
- #define [MTML_DEVICE_NAME_BUFFER_SIZE](#) 32
- #define [MTML_DEVICE_UUID_BUFFER_SIZE](#) 48
- #define [MTML_DEVICE_VBIOS_VERSION_BUFFER_SIZE](#) 64
- #define [MTML_DEVICE_PATH_BUFFER_SIZE](#) 64
- #define [MTML_DEVICE_PCI_SBDF_BUFFER_SIZE](#) 32
- #define [MTML_VIRT_TYPE_ID_BUFFER_SIZE](#) 16
- #define [MTML_VIRT_TYPE_CLASS_BUFFER_SIZE](#) 32
- #define [MTML_VIRT_TYPE_NAME_BUFFER_SIZE](#) 32
- #define [MTML_VIRT_TYPE_API_BUFFER_SIZE](#) 16
- #define [MTML_DEVICE_PCI_BUS_ID_FMT](#) "%08X:%02X:%02X.0"

Enumerations

- enum [MtmlReturn](#) {
 [MTML_SUCCESS](#) = 0 ,
 [MTML_ERROR_DRIVER_NOT_LOADED](#) ,
 [MTML_ERROR_DRIVER_FAILURE](#) ,
 [MTML_ERROR_INVALID_ARGUMENT](#) ,
 [MTML_ERROR_NOT_SUPPORTED](#) ,
 [MTML_ERROR_NO_PERMISSION](#) ,
 [MTML_ERROR_INSUFFICIENT_SIZE](#) ,
 [MTML_ERROR_NOT_FOUND](#) ,
 [MTML_ERROR_INSUFFICIENT_MEMORY](#) ,
 [MTML_ERROR_UNKNOWN](#) = 999 }
- enum [MtmlBrandType](#) {
 [MTML_BRAND_MTT](#) = 0 ,
 [MTML_BRAND_UNKNOWN](#) ,
 [MTML_BRAND_COUNT](#) }
- enum [MtmlCodecType](#)
- enum [MtmlVirtRole](#) {
 [MTML_VIRT_ROLE_NONE](#) ,
 [MTML_VIRT_ROLE_HOST_VIRTDEVICE](#) ,
 [MTML_VIRT_ROLE_COUNT](#) }

5.1.1 Detailed Description

5.1.2 Macro Definition Documentation

5.1.2.1 MTML_DEVICE_NAME_BUFFER_SIZE

```
#define MTML_DEVICE_NAME_BUFFER_SIZE 32
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the device name string (including a null terminator).

5.1.2.2 MTML_DEVICE_PATH_BUFFER_SIZE

```
#define MTML_DEVICE_PATH_BUFFER_SIZE 64
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the device path string (including a null terminator).

5.1.2.3 MTML_DEVICE_PCI_BUS_ID_FMT

```
#define MTML_DEVICE_PCI_BUS_ID_FMT "%08X:%02X:%02X.0"
```

Format string for PCI BUS ID.

5.1.2.4 MTML_DEVICE_PCI_SBDF_BUFFER_SIZE

```
#define MTML_DEVICE_PCI_SBDF_BUFFER_SIZE 32
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the PCI SBDF string of a device (including a null terminator).

5.1.2.5 MTML_DEVICE_UUID_BUFFER_SIZE

```
#define MTML_DEVICE_UUID_BUFFER_SIZE 48
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the UUID string of a device (including a null terminator).

5.1.2.6 MTML_DEVICE_VBIOS_VERSION_BUFFER_SIZE

```
#define MTML_DEVICE_VBIOS_VERSION_BUFFER_SIZE 64
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the VBIOS version string (including a null terminator).

5.1.2.7 MTML_DRIVER_VERSION_BUFFER_SIZE

```
#define MTML_DRIVER_VERSION_BUFFER_SIZE 80
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the driver version string (including a null terminator).

5.1.2.8 MTML_LIBRARY_VERSION_BUFFER_SIZE

```
#define MTML_LIBRARY_VERSION_BUFFER_SIZE 32
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the MTML library version string (including a null terminator).

5.1.2.9 MTML_VIRT_TYPE_API_BUFFER_SIZE

```
#define MTML_VIRT_TYPE_API_BUFFER_SIZE 16
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the API string of a virtualization type (including a null terminator).

5.1.2.10 MTML_VIRT_TYPE_CLASS_BUFFER_SIZE

```
#define MTML_VIRT_TYPE_CLASS_BUFFER_SIZE 32
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the class string of a virtualization type (including a null terminator).

5.1.2.11 MTML_VIRT_TYPE_ID_BUFFER_SIZE

```
#define MTML_VIRT_TYPE_ID_BUFFER_SIZE 16
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the ID string of a virtualization type (including a null terminator).

5.1.2.12 MTML_VIRT_TYPE_NAME_BUFFER_SIZE

```
#define MTML_VIRT_TYPE_NAME_BUFFER_SIZE 32
```

Recommended buffer size in bytes that is guaranteed to be enough to hold the name string of a virtualization type (including a null terminator).

5.1.3 Enumeration Type Documentation

5.1.3.1 MtmlBrandType

```
enum MtmlBrandType
```

The brand of the device.

Enumerator

MTML_BRAND_MTT	SUDI series.
MTML_BRAND_UNKNOWN	An unknown brand.
MTML_BRAND_COUNT	The number of brands.

5.1.3.2 MtmlCodecType

enum `MtmlCodecType`

The video codec types.

5.1.3.3 MtmlReturn

enum `MtmlReturn`

Return values for MTML API calls.

Enumerator

MTML_SUCCESS	The operation was successful.
MTML_ERROR_DRIVER_NOT_LOADED	The Moore Threads driver is not loaded.
MTML_ERROR_DRIVER_FAILURE	Access to the driver failed.
MTML_ERROR_INVALID_ARGUMENT	One or more supplied arguments are invalid.
MTML_ERROR_NOT_SUPPORTED	The requested operation is not available on the target device.
MTML_ERROR_NO_PERMISSION	The current user does not have permission to operate.
MTML_ERROR_INSUFFICIENT_SIZE	The space allocated by the user is insufficient to hold the requested data.
MTML_ERROR_NOT_FOUND	A query to find a resource was unsuccessful.
MTML_ERROR_INSUFFICIENT_MEMORY	There is not enough system memory to finish the operation.
MTML_ERROR_UNKNOWN	An internal unspecified error occurred.

5.1.3.4 MtmlVirtRole

enum `MtmlVirtRole`

The role of a virtualized device.

Enumerator

MTML_VIRT_ROLE_NONE	The device is not a virtual device or its virtualization role is unknown.
MTML_VIRT_ROLE_HOST_VIRTDEVICE	The device is a virtual device instantiated from a virtualization type of a physical device. This role is only reported in the host OS.

Enumerator

MTML_VIRT_ROLE_COUNT	The number of virtualization roles.
----------------------	-------------------------------------

5.2 Functional Structure Definitions

Classes

- struct [MtmlPciInfo](#)
- struct [MtmlCodecUtil](#)
- struct [MtmlCodecSessionMetrics](#)
- struct [MtmlVirtType](#)
- struct [MtmlDeviceProperty](#)

5.2.1 Detailed Description

5.3 Opaque Data Structure Definitions

Typedefs

- typedef struct [MtmlGpu](#) [MtmlGpu](#)
- typedef struct [MtmlMemory](#) [MtmlMemory](#)
- typedef struct [MtmlVpu](#) [MtmlVpu](#)
- typedef struct [MtmlDevice](#) [MtmlDevice](#)
- typedef struct [MtmlSystem](#) [MtmlSystem](#)
- typedef struct [MtmlLibrary](#) [MtmlLibrary](#)

5.3.1 Detailed Description

5.3.2 Typedef Documentation

5.3.2.1 MtmlDevice

```
typedef struct MtmlDevice MtmlDevice
```

Device opaque data structure.

5.3.2.2 MtmlGpu

```
typedef struct MtmlGpu MtmlGpu
```

GPU opaque data structure.

5.3.2.3 MtmlLibrary

```
typedef struct MtmlLibrary MtmlLibrary
```

Library opaque data structure.

5.3.2.4 MtmlMemory

```
typedef struct MtmlMemory MtmlMemory
```

Memory opaque data structure.

5.3.2.5 MtmlSystem

```
typedef struct MtmlSystem MtmlSystem
```

System opaque data structure.

5.3.2.6 MtmlVpu

```
typedef struct MtmlVpu MtmlVpu
```

VPU opaque data structure.

5.4 Library Functions

Functions

- **MtmlReturn** MTML_API **mtmlLibraryInit** (MtmlLibrary **lib)
- **MtmlReturn** MTML_API **mtmlLibraryShutDown** (MtmlLibrary *lib)
- **MtmlReturn** MTML_API **mtmlLibraryGetVersion** (const MtmlLibrary *lib, char *version, unsigned int length)
- **MtmlReturn** MTML_API **mtmlLibraryInitSystem** (const MtmlLibrary *lib, MtmlSystem **sys)
- **MtmlReturn** MTML_API **mtmlLibraryFreeSystem** (MtmlSystem *sys)
- **MtmlReturn** MTML_API **mtmlLibraryCountDevice** (const MtmlLibrary *lib, unsigned int *count)
- **MtmlReturn** MTML_API **mtmlLibraryInitDeviceByIndex** (const MtmlLibrary *lib, unsigned int index, MtmlDevice **dev)
- **MtmlReturn** MTML_API **mtmlLibraryInitDeviceByUuid** (const MtmlLibrary *library, const char *uuid, MtmlDevice **dev)
- **MtmlReturn** MTML_API **mtmlLibraryFreeDevice** (MtmlDevice *dev)

5.4.1 Detailed Description

5.4.2 Function Documentation

5.4.2.1 mtmlLibraryCountDevice()

```
MtmlReturn MTML_API mtmlLibraryCountDevice (  
    const MtmlLibrary * lib,  
    unsigned int * count )
```

Retrieves the number of devices that can be accessed by the library opaque object.

Parameters

<i>lib</i>	[in] The pointer to the library opaque object.
<i>count</i>	[out] The reference in which to return the number of accessible devices.

Returns

- [MTML_SUCCESS](#) if *count* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *count* or *lib* is NULL.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.4.2.2 mtmlLibraryFreeDevice()

```
MtmlReturn MTML_API mtmlLibraryFreeDevice (  
    MtmlDevice * dev )
```

Releases the resources managed by the device opaque object and makes it unusable.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the device opaque object that shall be freed.
------------	---

Returns

- [MTML_SUCCESS](#) if *dev* has been freed.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is a Virtual Device.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.4.2.3 mtmlLibraryFreeSystem()

```
MtmlReturn MTML_API mtmlLibraryFreeSystem (  
    MtmlSystem * sys )
```

Releases the resources managed by the MtmlSystem opaque object and makes it unusable.

For all products.

Parameters

<i>sys</i>	[in] The pointer to the system opaque object that shall be freed.
------------	---

Returns

- [MTML_SUCCESS](#) if *sys* has been freed.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *sys* is NULL.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.4.2.4 mtmlLibraryGetVersion()

```
MtmlReturn MTML_API mtmlLibraryGetVersion (
    const MtmlLibrary * lib,
    char * version,
    unsigned int length )
```

Retrieves the version of the specified MtmlLibrary opaque data.

For all products.

The version identifier is an alphanumeric string and null-terminator guaranteed.

Parameters

<i>lib</i>	[in] The pointer to library opaque data.
<i>version</i>	[out] The reference in which to return the version identifier.
<i>length</i>	[in] The length of the buffer pointed by <i>version</i> . See MTML_LIBRARY_VERSION_BUFFER_SIZE

Returns

- [MTML_SUCCESS](#) if *version* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *version* or *lib* is NULL.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *length* is too small to hold the version string.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.4.2.5 mtmlLibraryInit()

```
MtmlReturn MTML_API mtmlLibraryInit (
    MtmlLibrary ** lib )
```

Initializes an library opaque object and prepares resources. Upon success, the initialized object can be used for further access to other functions. An library opaque object is the top-level entry-point from which user code can access other functions provided by the MTML library.

For all products.

The output data is allocated internally by the MTML library. Use [mtmlLibraryShutDown\(\)](#) to release its resources when it is not needed any longer.

Parameters

<i>lib</i>	[out] A double pointer to the library opaque object that is allocated and initialized.
------------	--

Warning

Providing this function with a *lib* that has already been initialized causes memory leak.

Returns

- [MTML_SUCCESS](#) if an library opaque object has been properly initialized.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *lib* is NULL.
- [MTML_ERROR_INSUFFICIENT_MEMORY](#) if the system is running out of memory.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.4.2.6 mtmlLibraryInitDeviceByIndex()

```
MtmlReturn MTML_API mtmlLibraryInitDeviceByIndex (
    const MtmlLibrary * lib,
    unsigned int index,
    MtmlDevice ** dev )
```

Initializes a device opaque object to represent a device that is designated by its index. The index ranges from (0) to (deviceCount - 1), where deviceCount is retrieved from [mtmlLibraryCountDevice\(\)](#). The output data is allocated internally by the MTML library. Use [mtmlLibraryFreeDevice\(\)](#) to release its resources when it is not needed any longer.

For all products.

The initialized device opaque object can be used with device-related functions.

Parameters

<i>lib</i>	[in] The pointer to the library opaque object.
<i>index</i>	[in] The index of the target device.
<i>dev</i>	[out] The double pointer to the device opaque object that shall be allocated and initialized.

Warning

Providing this function with a *dev* that has already been initialized causes memory leak.

Returns

- [MTML_SUCCESS](#) if *device* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *index* is invalid, or either *lib* or *dev* is NULL.

- [MTML_ERROR_NOT_FOUND](#) if no device with the specified *index* is found.
- [MTML_ERROR_INSUFFICIENT_MEMORY](#) if the system is running out of memory.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

See also

[mtmlLibraryCountDevice](#)

5.4.2.7 mtmlLibraryInitDeviceByUuid()

```
MtmlReturn MTML_API mtmlLibraryInitDeviceByUuid (
    const MtmlLibrary * library,
    const char * uuid,
    MtmlDevice ** dev )
```

Initializes a device opaque object to represent a device that is designated by its UUID. See [mtmlDeviceGetUUID\(\)](#) for more information about the device's UUID. The output data is allocated internally by the MTML library. Use [mtmlLibraryFreeDevice\(\)](#) to release its resources when it is not needed any longer.

For all products.

The initialized device opaque object can be used with device-related functions.

Parameters

<i>lib</i>	[in] The pointer to the library opaque object.
<i>uuid</i>	[in] The UUID of the target device.
<i>dev</i>	[out] The double pointer to the device opaque object that shall be allocated and initialized.

Warning

Providing this function with a *dev* that has already been initialized causes memory leak.

Returns

- [MTML_SUCCESS](#) if *device* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *library*, *uuid*, or *dev* is NULL.
- [MTML_ERROR_NOT_FOUND](#) if no device with the specified *uuid* is found.
- [MTML_ERROR_INSUFFICIENT_MEMORY](#) if the system is running out of memory.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.4.2.8 mtmlLibraryInitSystem()

```
MtmlReturn MTML_API mtmlLibraryInitSystem (
    const MtmlLibrary * lib,
    MtmlSystem ** sys )
```

Initializes a MtmlSystem opaque pointer that is bound to a library opaque object. The output data is allocated internally by the MTML library. [mtmlLibraryFreeSystem\(\)](#) to release its resource when it is not needed any longer.

For all products.

The initialized MtmlSystem opaque object can be used with system-related functions.

Parameters

<i>lib</i>	[in] The pointer to library opaque data.
<i>sys</i>	[out] The double pointer to the system opaque data that shall be allocated and initialized.

Warning

Providing this function with a *sys* that has already been initialized causes memory leak.

Returns

- [MTML_SUCCESS](#) if *sys* has been initialized.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *lib* or *sys* is NULL.
- [MTML_ERROR_INSUFFICIENT_MEMORY](#) if the system is running out of memory.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.4.2.9 mtmlLibraryShutDown()

```
MtmlReturn MTML_API mtmlLibraryShutDown (
    MtmlLibrary * lib )
```

Shuts down a library opaque object that is previously initialized by [mtmlLibraryInit\(\)](#) and releases its resources. This opaque object cannot be used with other functions any longer unless being initialized again.

For all products.

Parameters

<i>lib</i>	[in] A pointer to the library opaque object that needs to be shut down.
------------	---

Returns

- [MTML_SUCCESS](#) if MTML has been properly shut down.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *lib* is NULL.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.5 System Functions

Functions

- [MtmlReturn](#) [MTML_API](#) [mtmlSystemGetDriverVersion](#) (const [MtmlSystem](#) *sys, char *version, unsigned int length)

5.5.1 Detailed Description

5.5.2 Function Documentation

5.5.2.1 mtmlSystemGetDriverVersion()

```
MtmlReturn MTML\_API mtmlSystemGetDriverVersion (  
    const MtmlSystem * sys,  
    char * version,  
    unsigned int length )
```

Retrieves the version of the driver that is installed on the current platform.

For all products.

The version identifier is an alphanumeric string.

Parameters

<i>sys</i>	[in] The pointer to the system opaque object.
<i>version</i>	[out] The reference in which to return the version identifier.
<i>length</i>	[in] The buffer size pointed by <i>version</i> . See MTML_DRIVER_VERSION_BUFFER_SIZE .

Returns

- [MTML_SUCCESS](#) if *version* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *sys* or *version* is NULL.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *length* is too small.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6 Device Functions

Functions

- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceInitGpu](#) (const [MtmlDevice](#) *dev, [MtmlGpu](#) **gpu)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceFreeGpu](#) ([MtmlGpu](#) *gpu)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceInitMemory](#) (const [MtmlDevice](#) *dev, [MtmlMemory](#) **mem)

- **MtmlReturn** MTML_API **mtmlDeviceFreeMemory** (**MtmlMemory** *mem)
- **MtmlReturn** MTML_API **mtmlDeviceInitVpu** (const **MtmlDevice** *dev, **MtmlVpu** **vpu)
- **MtmlReturn** MTML_API **mtmlDeviceFreeVpu** (**MtmlVpu** *vpu)
- **MtmlReturn** MTML_API **mtmlDeviceGetIndex** (const **MtmlDevice** *dev, unsigned int *index)
- **MtmlReturn** MTML_API **mtmlDeviceGetUUID** (const **MtmlDevice** *dev, char *uuid, unsigned int length)
- **MtmlReturn** MTML_API **mtmlDeviceGetBrand** (const **MtmlDevice** *dev, **MtmlBrandType** *type)
- **MtmlReturn** MTML_API **mtmlDeviceGetName** (const **MtmlDevice** *dev, char *name, unsigned int length)
- **MtmlReturn** MTML_API **mtmlDeviceGetPciInfo** (const **MtmlDevice** *dev, **MtmlPciInfo** *pci)
- **MtmlReturn** MTML_API **mtmlDeviceGetPowerUsage** (const **MtmlDevice** *dev, unsigned int *power)
- **MtmlReturn** MTML_API **mtmlDeviceGetGpuPath** (const **MtmlDevice** *dev, char *path, unsigned int length)
- **MtmlReturn** MTML_API **mtmlDeviceGetPrimaryPath** (const **MtmlDevice** *dev, char *path, unsigned int length)
- **MtmlReturn** MTML_API **mtmlDeviceGetRenderPath** (const **MtmlDevice** *dev, char *path, unsigned int length)
- **MtmlReturn** MTML_API **mtmlDeviceGetVbiosVersion** (const **MtmlDevice** *dev, char *version, unsigned int length)
- **MtmlReturn** MTML_API **mtmlDeviceGetProperty** (const **MtmlDevice** *dev, **MtmlDeviceProperty** *prop)

5.6.1 Detailed Description

5.6.2 Function Documentation

5.6.2.1 mtmlDeviceFreeGpu()

```
MtmlReturn MTML_API mtmlDeviceFreeGpu (
    MtmlGpu * gpu )
```

Releases the resources managed by the GPU opaque object and makes it unusable.

- For all products.

Parameters

<i>gpu</i>	[in] The pointer to the GPU opaque object that shall be freed.
------------	--

Returns

- **MTML_SUCCESS** if *gpu* has been freed.
- **MTML_ERROR_INVALID_ARGUMENT** if *gpu* is NULL.
- **MTML_ERROR_UNKNOWN** if any unexpected error occurred.

5.6.2.2 mtmlDeviceFreeMemory()

```
MtmlReturn MTML_API mtmlDeviceFreeMemory (
    MtmlMemory * mem )
```

Releases the resources managed by the memory opaque object and makes it unusable.

For all products.

Parameters

<i>mem</i>	[in] The pointer to the memory opaque object that shall be freed.
------------	---

Returns

- [MTML_SUCCESS](#) if *mem* has been freed.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *mem* is NULL.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.3 mtmlDeviceFreeVpu()

```
MtmlReturn MTML_API mtmlDeviceFreeVpu (  
    MtmlVpu * vpu )
```

Releases the resources managed by the VPU opaque object and makes it unusable.

For all products.

Parameters

<i>vpu</i>	[in] The pointer to the VPU opaque object that shall be freed.
------------	--

Returns

- [MTML_SUCCESS](#) if *vpu* has been freed.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *vpu* is NULL.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.4 mtmlDeviceGetBrand()

```
MtmlReturn MTML_API mtmlDeviceGetBrand (  
    const MtmlDevice * dev,  
    MtmlBrandType * type )
```

Retrieves the brand of a device.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>type</i>	[out] The reference in which to return the product brand type. See MtmlBrandType .

Returns

- [MTML_SUCCESS](#) if *name* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* is invalid or *type* is NULL.
- [MTML_ERROR_UNKNOWN](#) on any unexpected error.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.

5.6.2.5 mtmlDeviceGetGpuPath()

```
MtmlReturn MTML_API mtmlDeviceGetGpuPath (
    const MtmlDevice * dev,
    char * path,
    unsigned int length )
```

Retrieves the GPU paths of a device.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>path</i>	[out] The reference in which to return the target device path.
<i>length</i>	[in] The size of buffer pointed by <i>path</i> . See MTML_DEVICE_PATH_BUFFER_SIZE .

Returns

- [MTML_SUCCESS](#) if *path* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *path* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is a virtual device.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *length* is too small.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.6 mtmlDeviceGetIndex()

```
MtmlReturn MTML_API mtmlDeviceGetIndex (
    const MtmlDevice * dev,
    unsigned int * index )
```

Retrieves the index associated with the specified device.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the device opaque object.
<i>index</i>	[out] The index of the target device.

Returns

- [MTML_SUCCESS](#) if *index* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *index* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* does not support to be accessed by *index*. For example, the host virtual device.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.7 mtmlDeviceGetName()

```
MtmlReturn MTML_API mtmlDeviceGetName (  
    const MtmlDevice * dev,  
    char * name,  
    unsigned int length )
```

Retrieves the name of a device.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>name</i>	[out] The reference in which to return the product name.
<i>length</i>	[in] The size of buffer pointed by <i>name</i> . See MTML_DEVICE_NAME_BUFFER_SIZE.

Returns

- [MTML_SUCCESS](#) if *name* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *name* is NULL.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *length* is too small.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is a virtual device.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.8 mtmlDeviceGetPciInfo()

```
MtmlReturn MTML_API mtmlDeviceGetPciInfo (  
    const MtmlDevice * dev,  
    MtmlPciInfo * pci )
```

Retrieves the PCI attributes of a device.

For all products.

See [MtmlPciInfo](#) for details on the available PCI information.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>pci</i>	[out] The reference in which to return the PCI info.

Returns

- [MTML_SUCCESS](#) if *pci* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *pci* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is a virtual device.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.9 mtmlDeviceGetPowerUsage()

```
MtmlReturn MTML_API mtmlDeviceGetPowerUsage (  
    const MtmlDevice * dev,  
    unsigned int * power )
```

Retrieves the power usage for a device in milliwatts (mW) and its associated circuitry.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>power</i>	[out] The reference in which to return the power usage information.

Returns

- [MTML_SUCCESS](#) if *power* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *power* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if the operation is not supported.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.10 mtmlDeviceGetPrimaryPath()

```
MtmlReturn MTML_API mtmlDeviceGetPrimaryPath (  
    const MtmlDevice * dev,  
    char * path,  
    unsigned int length )
```

Retrieves the primary paths of a device.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>path</i>	[out] The reference in which to return the target device path.
<i>length</i>	[in] The size of buffer pointed by <i>path</i> . See MTML_DEVICE_PATH_BUFFER_SIZE .

Returns

- [MTML_SUCCESS](#) if *path* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *path* is NULL.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *length* is too small.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is a virtual device.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.11 mtmlDeviceGetProperty()

```
MtmlReturn MTML_API mtmlDeviceGetProperty (  
    const MtmlDevice * dev,  
    MtmlDeviceProperty * prop )
```

Retrieves the properties of a device.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>prop</i>	[out] The reference in which to return the target device properties.

Returns

- [MTML_SUCCESS](#) if *prop* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *prop* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.12 mtmlDeviceGetRenderPath()

```
MtmlReturn MTML_API mtmlDeviceGetRenderPath (  
    const MtmlDevice * dev,  
    char * path,  
    unsigned int length )
```

Retrieves the render paths of a device.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>path</i>	[out] The reference in which to return the target device path.
<i>length</i>	[in] The size of buffer pointed by <i>path</i> . See MTML_DEVICE_PATH_BUFFER_SIZE .

Returns

- [MTML_SUCCESS](#) if *path* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *path* is NULL.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *length* is too small.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is a virtual device.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.13 mtmIDeviceGetUUID()

```
MtmlReturn MTML_API mtmIDeviceGetUUID (
    const MtmlDevice * dev,
    char * uuid,
    unsigned int length )
```

Retrieves the UUID of a specified device. The UUID is a hexadecimal string in the form of xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx, where each 'x' is an ASCII character that represents a hexadecimal digit. The UUID is globally unique for every single device thus can be used to identify different devices physically.

For all products.

Parameters

<i>dev</i>	[in] The identifier of the target device.
<i>uuid</i>	[out] The reference in which to return the UUID.
<i>length</i>	[in] The size of buffer pointed by <i>uuid</i> . See MTML_DEVICE_UUID_BUFFER_SIZE .

Returns

- [MTML_SUCCESS](#) if *uuid* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *uuid* is NULL.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *length* is too small.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.14 mtmIDeviceGetVbiosVersion()

```
MtmlReturn MTML_API mtmIDeviceGetVbiosVersion (
    const MtmlDevice * dev,
```

```
char * version,  
unsigned int length )
```

Retrieves the version of the device's VBIOS.

For all products.

The version identifier is an alphanumeric string.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>version</i>	[out] The reference in which to return the version identifier.
<i>length</i>	[in] The size of buffer pointed by <i>version</i> . See MTML_DEVICE_VBIOS_VERSION_BUFFER_SIZE .

Returns

- [MTML_SUCCESS](#) if *version* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *version* is NULL.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *length* is too small.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is a virtual device.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.15 mtmIDeviceInitGpu()

```
MtmlReturn MTML_API mtmIDeviceInitGpu (  
    const MtmlDevice * dev,  
    MtmlGpu ** gpu )
```

Initializes a GPU opaque object to represent a specific graphic core on the target device that is designated by its index. The output data is allocated internally by the MTML library. Use [mtmIDeviceFreeGpu\(\)](#) to release its resources when it is not needed any longer.

For all products.

The initialized GPU opaque object can be used with GPU-related functions.

Parameters

<i>dev</i>	[in] The pointer to the device opaque object.
<i>gpu</i>	[out] The double pointer to the GPU opaque object that shall be allocated and initialized.

Warning

Providing this function with a *gpu* that has already been initialized causes memory leak.

Returns

- [MTML_SUCCESS](#) if *gpu* has been initialized successfully.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *gpu* is NULL.
- [MTML_ERROR_INSUFFICIENT_MEMORY](#) if the system is running out of memory.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.16 mtmlDeviceInitMemory()

```
MtmlReturn MTML_API mtmlDeviceInitMemory (  
    const MtmlDevice * dev,  
    MtmlMemory ** mem )
```

Initializes a memory opaque object to represent the memory on the target device. The output data is allocated internally by the MTML library. Use [mtmlDeviceFreeMemory\(\)](#) to release its resources when it is not needed any longer.

For all products.

The initialized memory opaque object can be used with memory-related functions.

Parameters

<i>dev</i>	[in] The pointer to the device opaque object.
<i>mem</i>	[out] The double pointer to the memory opaque object that shall be allocated and initialized.

Warning

Providing this function with a *mem* that has already been initialized causes memory leak.

Returns

- [MTML_SUCCESS](#) if *mem* has been initialized successfully.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *mem* is NULL.
- [MTML_ERROR_INSUFFICIENT_MEMORY](#) if the system is running out of memory.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.6.2.17 mtmlDeviceInitVpu()

```
MtmlReturn MTML_API mtmlDeviceInitVpu (  
    const MtmlDevice * dev,  
    MtmlVpu ** vpu )
```

Initializes a VPU opaque object to represent the video codec on the target device. The output data is allocated internally by the MTML library. Use [mtmlDeviceFreeVpu\(\)](#) to release its resources when it is not needed any longer.

For all products.

The initialized VPU opaque object can be used with codec-related functions.

Parameters

<i>dev</i>	[in] The pointer to the device opaque object.
<i>vpu</i>	[out] The double pointer to the memory opaque object that shall be allocated and initialized.

Warning

Providing this function with a *vpu* that has already been initialized causes memory leak.

Returns

- [MTML_SUCCESS](#) if *vpu* has been successfully initialized.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *vpu* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_INSUFFICIENT_MEMORY](#) if the system is running out of memory.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7 Device Virtualization Functions

Functions

- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceCountSupportedVirtTypes](#) (const [MtmlDevice](#) *dev, unsigned int *count)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceGetSupportedVirtTypes](#) (const [MtmlDevice](#) *dev, [MtmlVirtType](#) *types, unsigned int count)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceCountAvailVirtTypes](#) (const [MtmlDevice](#) *dev, unsigned int *count)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceGetAvailVirtTypes](#) (const [MtmlDevice](#) *dev, [MtmlVirtType](#) *types, unsigned int count)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceCountAvailVirtDevices](#) (const [MtmlDevice](#) *dev, const [MtmlVirtType](#) *type, unsigned int *count)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceCountActiveVirtDevices](#) (const [MtmlDevice](#) *dev, unsigned int *count)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceGetActiveVirtDeviceUuids](#) (const [MtmlDevice](#) *dev, char *uuids, unsigned int entryLength, unsigned int entryCount)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceInitVirtDevice](#) (const [MtmlDevice](#) *dev, const char *uuid, [MtmlDevice](#) **virtDev)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceFreeVirtDevice](#) ([MtmlDevice](#) *virtDev)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceGetVirtType](#) (const [MtmlDevice](#) *virtDev, [MtmlVirtType](#) *type)
- [MtmlReturn](#) [MTML_API](#) [mtmlDeviceGetPhyDeviceUuid](#) (const [MtmlDevice](#) *virtDev, char *uuid, unsigned int length)

5.7.1 Detailed Description

5.7.2 Function Documentation

5.7.2.1 [mtmlDeviceCountActiveVirtDevices\(\)](#)

```
MtmlReturn MTML\_API mtmlDeviceCountActiveVirtDevices (  
    const MtmlDevice * dev,  
    unsigned int * count )
```

Gets the number of active virtual devices created on a virtualizable device. A virtual device is active if it has been allocated resources as per its virtualization type. This function is ONLY applicable to a device that supports to be virtualized. Refer to [MtmlDeviceProperty.virtCap](#) to determine whether a device is virtualizable.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>count</i>	[out] The reference in which to return the number of active virtual devices.

Returns

- [MTML_SUCCESS](#) if *count* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *count* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is non-virtualizable.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7.2.2 mtmIDeviceCountAvailVirtDevices()

```
MtmlReturn MTML_API mtmIDeviceCountAvailVirtDevices (  
    const MtmlDevice * dev,  
    const MtmlVirtType * type,  
    unsigned int * count )
```

Gets the number of virtual devices that can be created from a specified virtualization type supported by a virtualizable device. This function is ONLY applicable to a device that supports to be virtualized. Refer to [MtmlDeviceProperty.virtCap](#) to determine whether a device is virtualizable.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>type</i>	[in] The specified virtualization type.
<i>count</i>	[out] The reference in which to return the number of available virtualization types.

Returns

- [MTML_SUCCESS](#) if *count* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *type* or *count* is NULL. or *type* is invalid or not supported.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is non-virtualizable.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7.2.3 mtmIDeviceCountAvailVirtTypes()

```
MtmlReturn MTML_API mtmIDeviceCountAvailVirtTypes (  
    const MtmlDevice * dev,  
    unsigned int * count )
```

Gets the number of virtualization types, from which a virtual device can be created, of a virtualizable device. This function is ONLY applicable to a device that supports to be virtualized. Refer to [MtmlDeviceProperty.virtCap](#) to determine whether a device is virtualizable.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>count</i>	[out] The reference in which to return the number of available virtualization types.

Returns

- [MTML_SUCCESS](#) if *count* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *count* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is non-virtualizable.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7.2.4 mtmlDeviceCountSupportedVirtTypes()

```
MtmlReturn MTML_API mtmlDeviceCountSupportedVirtTypes (  
    const MtmlDevice * dev,  
    unsigned int * count )
```

Gets the number of all virtualization types supported by a virtualizable device. This function is ONLY applicable to a device that supports to be virtualized. Refer to [MtmlDeviceProperty.virtCap](#) to determine whether a device is virtualizable.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>count</i>	[out] The reference in which to return the number of supported virtualization types.

Returns

- [MTML_SUCCESS](#) if *count* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *count* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is non-virtualizable.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7.2.5 mtmDeviceFreeVirtDevice()

```
MtmlReturn MTML_API mtmDeviceFreeVirtDevice (
    MtmlDevice * virtDev )
```

Releases the resources managed by the opaque object of a virtual device and makes it unusable. This function is ONLY applicable to a device that supports to be virtualized. Refer to [MtmlDeviceProperty.virtCap](#) to determine whether a device is virtualizable.

For all products.

Parameters

<i>virtDev</i>	[in] The pointer to a device opaque object that shall be freed.
----------------	---

Returns

- [MTML_SUCCESS](#) if *virtDev* has been freed.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *virtDev* is NULL or not a virtDev.
- [MTML_ERROR_NOT_SUPPORTED](#) if *virtDev* is not a Virtual Device.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7.2.6 mtmDeviceGetActiveVirtDeviceUuids()

```
MtmlReturn MTML_API mtmDeviceGetActiveVirtDeviceUuids (
    const MtmlDevice * dev,
    char * uuids,
    unsigned int entryLength,
    unsigned int entryCount )
```

Gets the list of UUIDs of all active virtual devices created on a virtualizable device. The UUID strings are output in the *uuids* parameter, which shall be a pointer to a continuous memory block whose size is enough to all content. The memory block is treated as a series of head-tail adjacent buffers (entry), while each of the buffer is *entryLength* bytes in size. The number of the buffers is defined by *entryCount*, thus the total size of the memory block pointed by *uuids* shall be equal or greater than *entryLength * entryCount*. Each buffer within the memory block is responsible for holding one UUID string with a null-terminator, so it's the caller's responsibility to ensure there are enough buffers to hold all UUID strings and also each buffer is enough in size to hold a single UUID string. If the *entryCount* is greater than the actual number of active virtual devices, the extra entries are filled with empty strings. A virtual device is active if it has been allocated resources as per its virtualization type. This function is ONLY applicable to a device that supports to be virtualized. Refer to [MtmlDeviceProperty.virtCap](#) to determine whether a device is virtualizable.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>uuids</i>	[out] A pointer to the memory block tha holds all ouput UUID strings.
<i>entryLength</i>	[in] The size of each entry (each buffer) in <i>uuids</i> .
<i>entryCount</i>	[in] The number of entries in <i>uuids</i> .

Returns

- [MTML_SUCCESS](#) if *uuids* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *uuids* is NULL.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *entrylength* or *entryCount* is too small.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is non-virtualizable.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7.2.7 mtmlDeviceGetAvailVirtTypes()

```
MtmlReturn MTML_API mtmlDeviceGetAvailVirtTypes (
    const MtmlDevice * dev,
    MtmlVirtType * types,
    unsigned int count )
```

Gets the list of virtualization types, from which a virtual device can be created, of a virtualizable device. This function is ONLY applicable to a device that supports to be virtualized. Refer to [MtmlDeviceProperty.virtCap](#) to determine whether a device is virtualizable.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>types</i>	[out] The reference in which to return the available virtualization types.
<i>count</i>	[in] The size of the array pointed by <i>types</i> .

Returns

- [MTML_SUCCESS](#) if *types* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *types* is NULL.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *count* is too small.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is non-virtualizable.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7.2.8 mtmlDeviceGetPhyDeviceUuid()

```
MtmlReturn MTML_API mtmlDeviceGetPhyDeviceUuid (
    const MtmlDevice * virtDev,
    char * uuid,
    unsigned int length )
```

Gets the physical device from which a virtual device is created. This function is ONLY applicable to virtual devices. A virtual device can be initialized by [mtmlDeviceInitVirtDevice\(\)](#) and always has a virtualization role of [MTML_VIRT_ROLE_HOST_VIRTDEVICE](#). Refer to [MtmlVirtRole](#) for more information.

For all products.

Parameters

<i>virtDev</i>	[in] The pointer to the target virtual device.
<i>uuid</i>	[out] The reference to which the UUID of the physical device is returned.
<i>length</i>	[in] The size of the buffer pointed by <i>uuid</i> , in bytes.

Returns

- [MTML_SUCCESS](#) if *version* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *virtDev* or *uuid* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if *virtDev* is not a virtual device.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *length* is too small.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7.2.9 mtmlDeviceGetSupportedVirtTypes()

```
MtmlReturn MTML_API mtmlDeviceGetSupportedVirtTypes (  
    const MtmlDevice * dev,  
    MtmlVirtType * types,  
    unsigned int count )
```

Gets the list of virtualization types supported by a virtualizable device. This function is ONLY applicable to a device that supports to be virtualized. Refer to [MtmlDeviceProperty.virtCap](#) to determine whether a device is virtualizable.

For all products.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>types</i>	[out] The reference in which to return the supported virtualization types.
<i>count</i>	[in] The size of the array pointed by <i>types</i> .

Returns

- [MTML_SUCCESS](#) if *types* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *types* is NULL.
- [MTML_ERROR_INSUFFICIENT_SIZE](#) if *count* is too small.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is non-virtualizable.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7.2.10 mtmlDeviceGetVirtType()

```
MtmlReturn MTML_API mtmlDeviceGetVirtType (
    const MtmlDevice * virtDev,
    MtmlVirtType * type )
```

Gets the virtualization type based on which a virtual device is created. This function is ONLY applicable to virtual devices. A virtual device can be initialized by [mtmlDeviceInitVirtDevice\(\)](#) and always has a virtualization role of [MTML_VIRT_ROLE_HOST_VIRTDEVICE](#). Refer to [MtmlVirtRole](#) for more information.

For all products.

Parameters

<i>virtDev</i>	[in] The pointer to the target virtual device.
<i>type</i>	[out] The reference to which the virtualization type is returned.

Returns

- [MTML_SUCCESS](#) if *type* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *virtDev* or *type* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if *virtDev* is not a Virtual Device.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.7.2.11 mtmlDeviceInitVirtDevice()

```
MtmlReturn MTML_API mtmlDeviceInitVirtDevice (
    const MtmlDevice * dev,
    const char * uuid,
    MtmlDevice ** virtDev )
```

Initializes an active virtual device that can be specified by the UUID from a virtualizable device. A virtual device is active if it has been allocated resources as per its virtualization type. This function is ONLY applicable to a device that supports to be virtualized. Refer to [MtmlDeviceProperty.virtCap](#) to determine whether a device is virtualizable. The output data is allocated internally by the MTML library. Use [mtmlDeviceFreeVirtDevice\(\)](#) to release its resources when it is not needed any longer.

For all products.

Warning

Providing this function with a *virtDev* that has already been initialized causes memory leak.

Parameters

<i>dev</i>	[in] The pointer to the target device.
<i>uuid</i>	[in] The UUID string of a virtual device.
<i>virtDev</i>	[out] The double pointer to the memory opaque object that shall be allocated and initialized.

Returns

- [MTML_SUCCESS](#) if *virtDev* has been initialized.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *dev* or *uuid* or *virtDev* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if *dev* is non-virtualizable.
- [MTML_ERROR_INSUFFICIENT_MEMORY](#) if the system is running out of memory.
- [MTML_ERROR_NOT_FOUND](#) if there is no Virtual Device found with *uuid*.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.8 GPU Functions

Functions

- [MtmlReturn](#) [MTML_API](#) [mtmlGpuGetUtilization](#) (const [MtmlGpu](#) *gpu, unsigned int *utilization)
- [MtmlReturn](#) [MTML_API](#) [mtmlGpuGetTemperature](#) (const [MtmlGpu](#) *gpu, unsigned int *temp)
- [MtmlReturn](#) [MTML_API](#) [mtmlGpuGetClock](#) (const [MtmlGpu](#) *gpu, unsigned int *clockMhz)
- [MtmlReturn](#) [MTML_API](#) [mtmlGpuGetMaxClock](#) (const [MtmlGpu](#) *gpu, unsigned int *clockMhz)

5.8.1 Detailed Description

5.8.2 Function Documentation

5.8.2.1 [mtmlGpuGetClock\(\)](#)

```
MtmlReturn MTML\_API mtmlGpuGetClock (  
    const MtmlGpu * gpu,  
    unsigned int * clockMhz )
```

Retrieves the current clock speed for the device's graphic core.

For all products.

Parameters

<i>gpu</i>	[in] The pointer to the GPU opaque object.
<i>clockMhz</i>	[out] The reference in which to return the clock speed in MHz.

Returns

- [MTML_SUCCESS](#) if *clockMhz* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *gpu* or *clockMhz* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.8.2.2 mtmlGpuGetMaxClock()

```
MtmlReturn MTML_API mtmlGpuGetMaxClock (
    const MtmlGpu * gpu,
    unsigned int * clockMhz )
```

Retrieves the maximum supported clock speed for the device's graphic core.

For all products.

Parameters

<i>gpu</i>	[in] The pointer to the GPU opaque object.
<i>clockMhz</i>	[out] The reference in which to return the clock speed in MHz.

Returns

- [MTML_SUCCESS](#) if *clockMhz* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *gpu* or *clockMhz* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.8.2.3 mtmlGpuGetTemperature()

```
MtmlReturn MTML_API mtmlGpuGetTemperature (
    const MtmlGpu * gpu,
    unsigned int * temp )
```

Retrieves the current temperature readings for the device's graphic core, in degrees Celsius.

Parameters

<i>gpu</i>	[in] The pointer to the GPU opaque object.
<i>temp</i>	[out] The reference in which to return the temperature reading.

Returns

- [MTML_SUCCESS](#) if *temp* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *gpu* or *temp* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.8.2.4 mtmLgpuGetUtilization()

```
MtmlReturn MTML_API mtmLgpuGetUtilization (
    const MtmlGpu * gpu,
    unsigned int * utilization )
```

Retrieves the current utilization rate for the device's graphic core.

For all products.

Parameters

<i>gpu</i>	[in] The pointer to the GPU opaque object.
<i>utilization</i>	[out] The reference in which to return the utilization information.

Returns

- [MTML_SUCCESS](#) if *utilization* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *gpu* or *utilization* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.9 Memory Functions

Functions

- [MtmlReturn](#) MTML_API [mtmLMemoryGetTotal](#) (const [MtmlMemory](#) *mem, unsigned long long *total)
- [MtmlReturn](#) MTML_API [mtmLMemoryGetUsed](#) (const [MtmlMemory](#) *mem, unsigned long long *used)
- [MtmlReturn](#) MTML_API [mtmLMemoryGetUtilization](#) (const [MtmlMemory](#) *mem, unsigned int *utilization)
- [MtmlReturn](#) MTML_API [mtmLMemoryGetClock](#) (const [MtmlMemory](#) *mem, unsigned int *clockMhz)
- [MtmlReturn](#) MTML_API [mtmLMemoryGetMaxClock](#) (const [MtmlMemory](#) *mem, unsigned int *clockMhz)

5.9.1 Detailed Description

5.9.2 Function Documentation

5.9.2.1 mtmLMemoryGetClock()

```
MtmlReturn MTML_API mtmLMemoryGetClock (
    const MtmlMemory * mem,
    unsigned int * clockMhz )
```

Retrieves the current clock speed for the memory of a device.

For all products.

Parameters

<i>mem</i>	[in] The pointer to the memory opaque object.
<i>clockMhz</i>	[out] The reference in which to return the clock speed in MHz.

Returns

- [MTML_SUCCESS](#) if *clockMhz* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *mem* or *clockMhz* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if the operation is not supported.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.9.2.2 mtmlMemoryGetMaxClock()

```
MtmlReturn MTML_API mtmlMemoryGetMaxClock (
    const MtmlMemory * mem,
    unsigned int * clockMhz )
```

Retrieves the maximum supported clock speed for the memory of a device.

For all products.

Parameters

<i>mem</i>	[in] The pointer to the memory opaque object.
<i>clockMhz</i>	[out] The reference in which to return the clock speed in MHz.

Returns

- [MTML_SUCCESS](#) if *clockMhz* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *mem* or *clockMhz* is NULL.
- [MTML_ERROR_NOT_SUPPORTED](#) if the operation is not supported.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.9.2.3 mtmlMemoryGetTotal()

```
MtmlReturn MTML_API mtmlMemoryGetTotal (
    const MtmlMemory * mem,
    unsigned long long * total )
```

Retrieves the amount of total memory available on the device, in bytes.

For all products.

Parameters

<i>mem</i>	[in] The pointer to the memory opaque object.
<i>total</i>	[out] The reference in which to return the total memory size.

Returns

- [MTML_SUCCESS](#) if *total* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *mem* is NULL or *total* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.9.2.4 mtmlMemoryGetUsed()

```
MtmlReturn MTML_API mtmlMemoryGetUsed (
    const MtmlMemory * mem,
    unsigned long long * used )
```

Retrieves the amount of used memory on the device, in bytes.

For all products.

Parameters

<i>mem</i>	[in] The pointer to the memory opaque object.
<i>used</i>	[out] The reference in which to return the used memory size.

Returns

- [MTML_SUCCESS](#) if *used* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *mem* is NULL or *used* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.9.2.5 mtmlMemoryGetUtilization()

```
MtmlReturn MTML_API mtmlMemoryGetUtilization (
    const MtmlMemory * mem,
    unsigned int * utilization )
```

Retrieves the current memory utilization rate for the device.

For all products.

Parameters

<i>mem</i>	[in] The pointer to the memory opaque object.
<i>utilization</i>	[out] The reference in which to return the utilization information.

Returns

- [MTML_SUCCESS](#) if *utilization* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *mem* or *utilization* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.10 VPU Functions

Functions

- [MtmlReturn](#) [MTML_API](#) [mtmlVpuGetUtilization](#) (const [MtmlVpu](#) *vpu, [MtmlCodecUtil](#) *utilization)
- [MtmlReturn](#) [MTML_API](#) [mtmlVpuGetClock](#) (const [MtmlVpu](#) *vpu, unsigned int *clockMhz)
- [MtmlReturn](#) [MTML_API](#) [mtmlVpuGetMaxClock](#) (const [MtmlVpu](#) *vpu, unsigned int *clockMhz)
- [MtmlReturn](#) [MTML_API](#) [mtmlVpuGetCodecCapacity](#) (const [MtmlVpu](#) *vpu, unsigned int *encodeCapacity, unsigned int *decodeCapacity)
- [MtmlReturn](#) [MTML_API](#) [mtmlVpuGetEncoderSessionStates](#) (const [MtmlVpu](#) *vpu, unsigned long long *states)
- [MtmlReturn](#) [MTML_API](#) [mtmlVpuGetEncoderSessionMetrics](#) (const [MtmlVpu](#) *vpu, unsigned int sessionId, [MtmlCodecSessionMetrics](#) *metrics)
- [MtmlReturn](#) [MTML_API](#) [mtmlVpuGetDecoderSessionStates](#) (const [MtmlVpu](#) *vpu, unsigned long long *states)
- [MtmlReturn](#) [MTML_API](#) [mtmlVpuGetDecoderSessionMetrics](#) (const [MtmlVpu](#) *vpu, unsigned int sessionId, [MtmlCodecSessionMetrics](#) *metrics)

5.10.1 Detailed Description

5.10.2 Function Documentation

5.10.2.1 mtmlVpuGetClock()

```
MtmlReturn MTML\_API mtmlVpuGetClock (
    const MtmlVpu * vpu,
    unsigned int * clockMhz )
```

Retrieves the current clock speed for the specified device's codec.

For all products.

Parameters

<i>vpu</i>	[in] The pointer to the VPU opaque object.
<i>clockMhz</i>	[out] The reference in which to return the clock speed in MHz.

Returns

- [MTML_SUCCESS](#) if *clockMhz* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *vpu* or *clockMhz* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.10.2.2 mtmIVpuGetCodecCapacity()

```
MtmlReturn MTML_API mtmIVpuGetCodecCapacity (  
    const MtmlVpu * vpu,  
    unsigned int * encodeCapacity,  
    unsigned int * decodeCapacity )
```

Retrieves the capacity, that is, maximum number of supported concurrent codec sessions, of the specified device's codec. A codec session is the processing of an independent video stream, which can be categorized into an encoder session and a decoder session. An encoder session refers to the video encoding processing, while a decoder session represents the decoding processing.

An encoder session can be referred by its ID (session ID), which ranges from 0 to *encodeCapacity* - 1. A decoder session can be referred by its ID (session ID), which ranges from 0 to *decodeCapacity* - 1.

For all products.

Parameters

<i>vpu</i>	[in] The pointer to the VPU opaque object.
<i>encodeCapacity</i>	[out] The reference in which to return the concurrent session capacity for encoding.
<i>decodeCapacity</i>	[out] The reference in which to return the concurrent session capacity for decoding.

Returns

- [MTML_SUCCESS](#) if *encodeCapacity* or *decodeCapacity* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *vpu* or *encodeCapacity* or *decodeCapacity* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.10.2.3 mtmIVpuGetDecoderSessionMetrics()

```
MtmlReturn MTML_API mtmIVpuGetDecoderSessionMetrics (  
    const MtmlVpu * vpu,
```

```
unsigned int sessionId,
MtmlCodecSessionMetrics * metrics )
```

Retrieves the metrics of the specified decoder session. If the session is not active (there is no system process attached to the session), the pid field of the output *metrics* struct is set to zero. Otherwise, the session is considered active and the output *metrics* struct is populated with valid data.

For all products.

Parameters

<i>vpu</i>	[in] The pointer to the VPU opaque object.
<i>sessionId</i>	[in] The ID of the session whose metrics shall be queried.
<i>metrics</i>	[out] The pointer to the output struct to hold the session metrics.

Returns

- **MTML_SUCCESS** if *metrics* has been set.
- **MTML_ERROR_INVALID_ARGUMENT** if *vpu* or *metrics* is NULL, or *sessionId* is invalid.
- **MTML_ERROR_DRIVER_FAILURE** if any error occurred when accessing the driver.
- **MTML_ERROR_UNKNOWN** if any unexpected error occurred.

5.10.2.4 mtmIVpuGetDecoderSessionStates()

```
MtmlReturn MTML_API mtmIVpuGetDecoderSessionStates (
    const MtmlVpu * vpu,
    unsigned long long * states )
```

Retrieves the state of each decoder session on a specified VPU. The output *states* is a bits pool, where each bit represents the state of a decoder session. The LSB of the *states* represents the state of the decoder session with ID 0, and so on. If a bit is 0, then the corresponding session is idle, otherwise, it's active. The decoder session capacity of a VPU can be retrieved from [mtmIVpuGetCodecCapacity\(\)](#), which can be used as the indication of how many bits are used in the *states*.

For all products.

Parameters

<i>vpu</i>	[in] The pointer to the VPU opaque object.
<i>states</i>	[out] The reference in which to return the states of decoder sessions.

Returns

- **MTML_SUCCESS** if *states* has been populated.
- **MTML_ERROR_INVALID_ARGUMENT** if *vpu* or *states* is NULL.
- **MTML_ERROR_DRIVER_FAILURE** if any error occurred when accessing the driver.
- **MTML_ERROR_UNKNOWN** if any unexpected error occurred.

5.10.2.5 mtmIVpuGetEncoderSessionMetrics()

```
MtmlReturn MTML_API mtmIVpuGetEncoderSessionMetrics (
    const MtmlVpu * vpu,
    unsigned int sessionId,
    MtmlCodecSessionMetrics * metrics )
```

Retrieves the metrics of the specified encoder session. If the session is not active (there is no system process attached to the session), the pid field of the output *metrics* struct is set to zero. Otherwise, the session is considered active and the output *metrics* struct is populated with valid data.

For all products.

Parameters

<i>vpu</i>	[in] The pointer to the VPU opaque object.
<i>sessionId</i>	[in] The ID of the session whose metrics shall be queried.
<i>metrics</i>	[out] The pointer to the output struct to hold the session metrics.

Returns

- [MTML_SUCCESS](#) if *metrics* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *vpu* or *metrics* is NULL, or *sessionId* is invalid.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.10.2.6 mtmIVpuGetEncoderSessionStates()

```
MtmlReturn MTML_API mtmIVpuGetEncoderSessionStates (
    const MtmlVpu * vpu,
    unsigned long long * states )
```

Retrieves the state of each encoder session on a specified VPU. The output *states* is a bits pool, where each bit represents the state of an encoder session. The LSB of the *states* represents the state of the encoder session with ID 0, and so on. If a bit is 0, then the corresponding session is idle, otherwise it's active. The encoder session capacity of a VPU can be retrieved from [mtmIVpuGetCodecCapacity\(\)](#) interface, which can be used as the indication of how many bits are used in the *states*.

For all products.

Parameters

<i>vpu</i>	[in] The pointer to the VPU opaque object.
<i>states</i>	[out] The reference in which to return the state of encoder sessions.

Returns

- [MTML_SUCCESS](#) if *states* has been populated.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *vpu* or *states* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.10.2.7 mtmIVpuGetMaxClock()

```
MtmlReturn MTML_API mtmIVpuGetMaxClock (
    const MtmlVpu * vpu,
    unsigned int * clockMhz )
```

Retrieves the maximum supported clock speed for the specified device's codec.

For all products.

Parameters

<i>vpu</i>	[in] The pointer to the VPU opaque object.
<i>clockMhz</i>	[out] The reference in which to return the clock speed in MHz.

Returns

- [MTML_SUCCESS](#) if *clockMhz* has been set.
- [MTML_ERROR_INVALID_ARGUMENT](#) if *vpu* or *clockMhz* is NULL.
- [MTML_ERROR_DRIVER_FAILURE](#) if any error occurred when accessing the driver.
- [MTML_ERROR_UNKNOWN](#) if any unexpected error occurred.

5.10.2.8 mtmIVpuGetUtilization()

```
MtmlReturn MTML_API mtmIVpuGetUtilization (
    const MtmlVpu * vpu,
    MtmlCodecUtil * utilization )
```

Retrieves the current utilization rates for the specified device's codec.

For all products.

Parameters

<i>vpu</i>	[in] The pointer to the VPU opaque object.
<i>utilization</i>	[out] The reference in which to return the utilization information.

Returns

- `MTML_SUCCESS` if *utilization* has been populated.
- `MTML_ERROR_INVALID_ARGUMENT` if *vpu* is NULL or *utilization* is NULL.
- `MTML_ERROR_DRIVER_FAILURE` if any error occurred when accessing the driver.
- `MTML_ERROR_UNKNOWN` if any unexpected error occurred.

Chapter 6

Class Documentation

6.1 MtmlPciInfo Struct Reference

```
#include <mtml.h>
```

Public Attributes

- char **sbdf** [[MTML_DEVICE_PCI_SBDF_BUFFER_SIZE](#)]
The tuple segment:bus:device.function PCI identifier (& NULL terminator).
- unsigned int **segment**
The PCI segment group(domain) on which the device's bus resides, 0 to 0xffffffff.
- unsigned int **bus**
The bus on which the device resides, 0 to 0xff.
- unsigned int **device**
The device ID on the bus, 0 to 31.
- unsigned int **pciDeviceId**
The combined 16-bit device ID and 16-bit vendor ID.
- unsigned int **pciSubsystemId**
The 32-bit sub system device ID.
- unsigned int **busWidth**
The bus width of the device.
- float **pciMaxSpeed**
The maximum link speed of the device. The unit is GT/s.
- float **pciCurSpeed**
The current link speed of the device. The unit is GT/s.
- unsigned int **pciMaxWidth**
The maximum link width of the device.
- unsigned int **pciCurWidth**
The current link width of the device.
- int **rsvd** [8]
Reserved for future extension.

6.1.1 Detailed Description

PCI information about a device.

The documentation for this struct was generated from the following file:

- `mtml.h`

6.2 MtmlCodecUtil Struct Reference

```
#include <mtml.h>
```

Public Attributes

- unsigned int **util**
Percent of time, during which the video codec was executed, over the past sample period.
- unsigned int **period**
The sampling period of time, in microseconds.
- int **rsvd** [4]
Reserved for future extension.

6.2.1 Detailed Description

Codec utilization percentage on a device.

The documentation for this struct was generated from the following file:

- `mtml.h`

6.3 MtmlCodecSessionMetrics Struct Reference

```
#include <mtml.h>
```

Public Attributes

- unsigned int **id**
The unique identifier of the code session.
- unsigned int **pid**
The process ID. 0 represents an idle session, otherwise the session is considered active.
- unsigned int **hResolution**
The horizontal resolution in pixels.
- unsigned int **vResolution**
The vertical resolution in pixels.
- unsigned int **frameRate**
The number of frames per second.
- unsigned int **bitRate**
The number of bits per second.
- unsigned int **latency**
The codec latency in microseconds.
- [MtmlCodecType](#) **codecType**
Type of codec. For example, H.265 and AV1.
- int **rsvd** [4]
Reserved for future extension.

6.3.1 Detailed Description

Codec session metrics.

The documentation for this struct was generated from the following file:

- `mtml.h`

6.4 MtmlVirtType Struct Reference

```
#include <mtml.h>
```

Public Attributes

- char **id** [[MTML_VIRT_TYPE_ID_BUFFER_SIZE](#)]
The ID of the virtualization type. For example, mtgpu-2008.
- char **name** [[MTML_VIRT_TYPE_NAME_BUFFER_SIZE](#)]
The name of the virtualization type. For example, mtgpu-8g.
- char **api** [[MTML_VIRT_TYPE_API_BUFFER_SIZE](#)]
The API type of the virtualization type. For example, vfio-pci.
- int **rsvd** [18]
Reserved for future extension.

6.4.1 Detailed Description

The type of virtualization that describes the specification of a virtualized device.

The documentation for this struct was generated from the following file:

- `mtml.h`

6.5 MtmlDeviceProperty Struct Reference

```
#include <mtml.h>
```

Public Attributes

- unsigned int **virtCap**: 1
- unsigned int **virtRole**: 3
- unsigned int **rsvd**: 28
Reserved for future extension.
- unsigned int **rsvd2**: 32
Reserved for future extension.

6.5.1 Detailed Description

The property of a device.

NOTE: A physical device that does not support virtualization (for example, S10) might still be able to be used in pass-through manner. For such device, its 'virtCap' field is '0 - non-virtualizable'.

6.5.2 Member Data Documentation

6.5.2.1 virtCap

```
unsigned int MtmlDeviceProperty::virtCap
```

This field indicates the virtualization capability of a device (whether a device supports to be virtualized) : 0 indicates non-virtualizable and 1 indicates virtualizable.

6.5.2.2 virtRole

```
unsigned int MtmlDeviceProperty::virtRole
```

This field indicates the role that this device is playing in a virtualization-supported environment. Refer to [MtmlVirtRole](#).

The documentation for this struct was generated from the following file:

- mtml.h

Chapter 7

File Documentation

7.1 mtml.h

```
1
31 #ifndef MTML_H_
32 #define MTML_H_
33
34 #ifdef __cplusplus
35 extern "C" {
36 #endif
37
38 #if defined(_MSC_VER)
39 #ifdef MTML_EXPORT
40 #define MTML_API __declspec(dllexport)
41 #else
42 #define MTML_API __declspec(dllimport)
43 #endif
44 #else
45 #define MTML_API __attribute__((visibility("default")))
46 #endif
47
126 /*****
130 /*****
131
135 #define MTML_LIBRARY_VERSION_BUFFER_SIZE          32
136
140 #define MTML_DRIVER_VERSION_BUFFER_SIZE           80
141
145 #define MTML_DEVICE_NAME_BUFFER_SIZE              32
146
150 #define MTML_DEVICE_UUID_BUFFER_SIZE              48
151
155 #define MTML_DEVICE_VBIOS_VERSION_BUFFER_SIZE     64
156
160 #define MTML_DEVICE_PATH_BUFFER_SIZE              64
161
165 #define MTML_DEVICE_PCI_SBDF_BUFFER_SIZE          32
166
170 #define MTML_VIRT_TYPE_ID_BUFFER_SIZE             16
171
175 #define MTML_VIRT_TYPE_CLASS_BUFFER_SIZE          32
176
180 #define MTML_VIRT_TYPE_NAME_BUFFER_SIZE           32
181
185 #define MTML_VIRT_TYPE_API_BUFFER_SIZE            16
186
190 #define MTML_DEVICE_PCI_BUS_ID_FMT                "%08X:%02X:%02X.0"
191
192
196 typedef enum {
197     MTML_SUCCESS = 0,
198     MTML_ERROR_DRIVER_NOT_LOADED,
199     MTML_ERROR_DRIVER_FAILURE,
200     MTML_ERROR_INVALID_ARGUMENT,
201     MTML_ERROR_NOT_SUPPORTED,
202     MTML_ERROR_NO_PERMISSION,
203     MTML_ERROR_INSUFFICIENT_SIZE,
204     MTML_ERROR_NOT_FOUND,
205     MTML_ERROR_INSUFFICIENT_MEMORY,
206     MTML_ERROR_UNKNOWN = 999
207 } MtmlReturn;
```

```
208
212 typedef enum {
213     MTML_BRAND_MTT = 0,
214     MTML_BRAND_UNKNOWN,
215
216     // Keep this on the last line.
217     MTML_BRAND_COUNT
218 } MtmlBrandType;
219
223 typedef enum {
224     MTML_CODEC_TYPE_AVC = 0,
225     MTML_CODEC_TYPE_VC1 = 1,
226     MTML_CODEC_TYPE_MPEG2 = 2,
227     MTML_CODEC_TYPE_MPEG4 = 3,
228     MTML_CODEC_TYPE_H263 = 4,
229     MTML_CODEC_TYPE_DIV3 = 5,
230     MTML_CODEC_TYPE_RV = 6,
231     MTML_CODEC_TYPE_AVS = 7,
232     MTML_CODEC_TYPE_THO = 9,
233     MTML_CODEC_TYPE_VP3 = 10,
234     MTML_CODEC_TYPE_VP8 = 11,
235     MTML_CODEC_TYPE_HEVC = 12,
236     MTML_CODEC_TYPE_VP9 = 13,
237     MTML_CODEC_TYPE_AVS2 = 14,
238     MTML_CODEC_TYPE_AV1 = 16,
239
240     // Keep this on the last line.
241     MTML_CODEC_TYPE_COUNT
242 } MtmlCodecType;
243
247 typedef enum {
248     MTML_VIRT_ROLE_NONE,
249     MTML_VIRT_ROLE_HOST_VIRTDEVICE,
250
251     // Keep this on the last line.
252     MTML_VIRT_ROLE_COUNT,
253 } MtmlVirtRole;
254
255
256 /*****
259 /*****
260
261
262
263 /*****
267 /*****
268
272 typedef struct {
273     char sbdf[MTML_DEVICE_PCI_SBDF_BUFFER_SIZE];
274     unsigned int segment;
275     unsigned int bus;
276     unsigned int device;
277     unsigned int pciDeviceId;
278     unsigned int pciSubsystemId;
279     unsigned int busWidth;
280     float pciMaxSpeed;
281     float pciCurSpeed;
282     unsigned int pciMaxWidth;
283     unsigned int pciCurWidth;
284     int rsvd[8];
285 } MtmlPciInfo;
286
290 typedef struct {
291     unsigned int util;
292     unsigned int period;
293     int rsvd[4];
294 } MtmlCodecUtil;
295
299 typedef struct {
300     unsigned int id;
301     unsigned int pid;
302     unsigned int hResolution;
303     unsigned int vResolution;
304     unsigned int frameRate;
305     unsigned int bitRate;
306     unsigned int latency;
307     MtmlCodecType codecType;
308     int rsvd[4];
309 } MtmlCodecSessionMetrics;
310
314 typedef struct {
315     char id[MTML_VIRT_TYPE_ID_BUFFER_SIZE];
316     char name[MTML_VIRT_TYPE_NAME_BUFFER_SIZE];
317     char api[MTML_VIRT_TYPE_API_BUFFER_SIZE];
318     int rsvd[18];
319 } MtmlVirtType;
320
328 typedef struct {
```

```
329     unsigned int virtCap : 1;
331     unsigned int virtRole : 3;
333     unsigned int rsvd : 28;
334     unsigned int rsvd2 : 32;
335 } MtmlDeviceProperty;
336
337 /*****/
340 /*****/
341
342
343
344 /*****/
348 /*****/
349
353 typedef struct MtmlGpu MtmlGpu;
354
358 typedef struct MtmlMemory MtmlMemory;
359
363 typedef struct MtmlVpu MtmlVpu;
364
368 typedef struct MtmlDevice MtmlDevice;
369
373 typedef struct MtmlSystem MtmlSystem;
374
378 typedef struct MtmlLibrary MtmlLibrary;
379
380 /*****/
383 /*****/
384
385
386
387 /*****/
391 /*****/
392
414 MtmlReturn MTML_API mtmlLibraryInit (MtmlLibrary **lib);
415
429 MtmlReturn MTML_API mtmlLibraryShutDown (MtmlLibrary *lib);
430
448 MtmlReturn MTML_API mtmlLibraryGetVersion (const MtmlLibrary *lib, char* version, unsigned int length);
449
470 MtmlReturn MTML_API mtmlLibraryInitSystem (const MtmlLibrary *lib, MtmlSystem **sys);
471
483 MtmlReturn MTML_API mtmlLibraryFreeSystem (MtmlSystem *sys);
484
496 MtmlReturn MTML_API mtmlLibraryCountDevice (const MtmlLibrary *lib, unsigned int *count);
497
525 MtmlReturn MTML_API mtmlLibraryInitDeviceByIndex (const MtmlLibrary *lib, unsigned int index, MtmlDevice
    **dev);
526
551 MtmlReturn MTML_API mtmlLibraryInitDeviceByUuid (const MtmlLibrary *library, const char *uuid, MtmlDevice
    **dev);
552
566 MtmlReturn MTML_API mtmlLibraryFreeDevice (MtmlDevice *dev);
567
568 /*****/
571 /*****/
572
573
574
575 /*****/
579 /*****/
580
598 MtmlReturn MTML_API mtmlSystemGetDriverVersion (const MtmlSystem *sys, char *version, unsigned int
    length);
599
600 /*****/
603 /*****/
604
605
606 /*****/
610 /*****/
611
632 MtmlReturn MTML_API mtmlDeviceInitGpu (const MtmlDevice *dev, MtmlGpu **gpu);
633
645 MtmlReturn MTML_API mtmlDeviceFreeGpu (MtmlGpu *gpu);
646
668 MtmlReturn MTML_API mtmlDeviceInitMemory (const MtmlDevice *dev, MtmlMemory **mem);
669
682 MtmlReturn MTML_API mtmlDeviceFreeMemory (MtmlMemory *mem);
683
706 MtmlReturn MTML_API mtmlDeviceInitVpu (const MtmlDevice *dev, MtmlVpu **vpu);
707
720 MtmlReturn MTML_API mtmlDeviceFreeVpu (MtmlVpu *vpu);
721
722
738 MtmlReturn MTML_API mtmlDeviceGetIndex (const MtmlDevice *dev, unsigned int *index);
739
```

```
759 MtmlReturn MTML_API mtmlDeviceGetUUID(const MtmlDevice *dev, char *uuid, unsigned int length);
760
775 MtmlReturn MTML_API mtmlDeviceGetBrand(const MtmlDevice *dev, MtmlBrandType *type);
776
793 MtmlReturn MTML_API mtmlDeviceGetName(const MtmlDevice *dev, char *name, unsigned int length);
794
812 MtmlReturn MTML_API mtmlDeviceGetPciInfo(const MtmlDevice *dev, MtmlPciInfo *pci);
813
828 MtmlReturn MTML_API mtmlDeviceGetPowerUsage(const MtmlDevice *dev, unsigned int *power);
829
846 MtmlReturn MTML_API mtmlDeviceGetGpuPath(const MtmlDevice* dev, char* path, unsigned int length);
847
865 MtmlReturn MTML_API mtmlDeviceGetPrimaryPath(const MtmlDevice* dev, char* path, unsigned int length);
866
884 MtmlReturn MTML_API mtmlDeviceGetRenderPath(const MtmlDevice* dev, char* path, unsigned int length);
885
905 MtmlReturn MTML_API mtmlDeviceGetVbiosVersion(const MtmlDevice* dev, char* version, unsigned int
length);
906
921 MtmlReturn MTML_API mtmlDeviceGetProperty(const MtmlDevice *dev, MtmlDeviceProperty *prop);
922
923
924 /*****/
927 /*****/
928
929
930
931 /*****/
935 /*****/
936
954 MtmlReturn MTML_API mtmlDeviceCountSupportedVirtTypes(const MtmlDevice *dev, unsigned int *count);
955
975 MtmlReturn MTML_API mtmlDeviceGetSupportedVirtTypes(const MtmlDevice *dev, MtmlVirtType *types, unsigned
int count);
976
994 MtmlReturn MTML_API mtmlDeviceCountAvailVirtTypes(const MtmlDevice *dev, unsigned int *count);
995
1015 MtmlReturn MTML_API mtmlDeviceGetAvailVirtTypes(const MtmlDevice *dev, MtmlVirtType *types, unsigned
int count);
1016
1036 MtmlReturn MTML_API mtmlDeviceCountAvailVirtDevices(const MtmlDevice *dev, const MtmlVirtType *type,
unsigned int *count);
1037
1056 MtmlReturn MTML_API mtmlDeviceCountActiveVirtDevices(const MtmlDevice *dev, unsigned int *count);
1057
1086 MtmlReturn MTML_API mtmlDeviceGetActiveVirtDeviceUuids(const MtmlDevice *dev, char *uuids, unsigned int
entryLength, unsigned int entryCount);
1087
1113 MtmlReturn MTML_API mtmlDeviceInitVirtDevice(const MtmlDevice *dev, const char *uuid, MtmlDevice
**virtDev);
1114
1130 MtmlReturn MTML_API mtmlDeviceFreeVirtDevice(MtmlDevice *virtDev);
1131
1150 MtmlReturn MTML_API mtmlDeviceGetVirtType(const MtmlDevice *virtDev, MtmlVirtType *type);
1151
1172 MtmlReturn MTML_API mtmlDeviceGetPhyDeviceUuid(const MtmlDevice *virtDev, char *uuid, unsigned int
length);
1173
1174 /*****/
1177 /*****/
1178
1179
1180
1181 /*****/
1185 /*****/
1186
1201 MtmlReturn MTML_API mtmlGpuGetUtilization(const MtmlGpu *gpu, unsigned int* utilization);
1202
1216 MtmlReturn MTML_API mtmlGpuGetTemperature(const MtmlGpu *gpu, unsigned int* temp);
1217
1218
1233 MtmlReturn MTML_API mtmlGpuGetClock(const MtmlGpu *gpu, unsigned int *clockMhz);
1234
1249 MtmlReturn MTML_API mtmlGpuGetMaxClock(const MtmlGpu *gpu, unsigned int *clockMhz);
1250
1251
1252 /*****/
1255 /*****/
1256
1257
1258
1259 /*****/
1263 /*****/
1264
1265
1280 MtmlReturn MTML_API mtmlMemoryGetTotal(const MtmlMemory *mem, unsigned long long *total);
1281
```

```
1296 MtmlReturn MTML_API mtmlMemoryGetUsed(const MtmlMemory *mem, unsigned long long *used);
1297
1312 MtmlReturn MTML_API mtmlMemoryGetUtilization(const MtmlMemory *mem, unsigned int *utilization);
1313
1329 MtmlReturn MTML_API mtmlMemoryGetClock(const MtmlMemory *mem, unsigned int *clockMhz);
1330
1346 MtmlReturn MTML_API mtmlMemoryGetMaxClock(const MtmlMemory *mem, unsigned int *clockMhz);
1347
1348
1349 /*****/
1352 /*****/
1353
1354
1355
1356 /*****/
1360 /*****/
1361
1376 MtmlReturn MTML_API mtmlVpuGetUtilization(const MtmlVpu *vpu, MtmlCodecUtil *utilization);
1377
1392 MtmlReturn MTML_API mtmlVpuGetClock(const MtmlVpu *vpu, unsigned int *clockMhz);
1393
1408 MtmlReturn MTML_API mtmlVpuGetMaxClock(const MtmlVpu *vpu, unsigned int *clockMhz);
1409
1431 MtmlReturn MTML_API mtmlVpuGetCodecCapacity(const MtmlVpu *vpu, unsigned int *encodeCapacity, unsigned
    int *decodeCapacity);
1432
1452 MtmlReturn MTML_API mtmlVpuGetEncoderSessionStates(const MtmlVpu *vpu, unsigned long long *states);
1453
1471 MtmlReturn MTML_API mtmlVpuGetEncoderSessionMetrics(const MtmlVpu *vpu, unsigned int sessionId,
    MtmlCodecSessionMetrics *metrics);
1472
1492 MtmlReturn MTML_API mtmlVpuGetDecoderSessionStates(const MtmlVpu *vpu, unsigned long long *states);
1493
1511 MtmlReturn MTML_API mtmlVpuGetDecoderSessionMetrics(const MtmlVpu *vpu, unsigned int sessionId,
    MtmlCodecSessionMetrics *metrics);
1512
1513
1514 /*****/
1517 /*****/
1518
1519
1520 #ifdef __cplusplus
1521 } // extern "C"
1522 #endif
1523
1524 #endif // MTML_H_
```

Index

Constant Definitions, 6

Constant definitions

- MTML_BRAND_COUNT, 9
- MTML_BRAND_MTT, 9
- MTML_BRAND_UNKNOWN, 9
- MTML_DEVICE_NAME_BUFFER_SIZE, 7
- MTML_DEVICE_PATH_BUFFER_SIZE, 7
- MTML_DEVICE_PCI_BUS_ID_FMT, 7
- MTML_DEVICE_PCI_SBDF_BUFFER_SIZE, 7
- MTML_DEVICE_UUID_BUFFER_SIZE, 7
- MTML_DEVICE_VBIOS_VERSION_BUFFER_SIZE, 7
- MTML_DRIVER_VERSION_BUFFER_SIZE, 7
- MTML_ERROR_DRIVER_FAILURE, 9
- MTML_ERROR_DRIVER_NOT_LOADED, 9
- MTML_ERROR_INSUFFICIENT_MEMORY, 9
- MTML_ERROR_INSUFFICIENT_SIZE, 9
- MTML_ERROR_INVALID_ARGUMENT, 9
- MTML_ERROR_NO_PERMISSION, 9
- MTML_ERROR_NOT_FOUND, 9
- MTML_ERROR_NOT_SUPPORTED, 9
- MTML_ERROR_UNKNOWN, 9
- MTML_LIBRARY_VERSION_BUFFER_SIZE, 8
- MTML_SUCCESS, 9
- MTML_VIRT_ROLE_COUNT, 10
- MTML_VIRT_ROLE_HOST_VIRTDEVICE, 9
- MTML_VIRT_ROLE_NONE, 9
- MTML_VIRT_TYPE_API_BUFFER_SIZE, 8
- MTML_VIRT_TYPE_CLASS_BUFFER_SIZE, 8
- MTML_VIRT_TYPE_ID_BUFFER_SIZE, 8
- MTML_VIRT_TYPE_NAME_BUFFER_SIZE, 8
- MtmlBrandType, 8
- MtmlCodecType, 9
- MtmlReturn, 9
- MtmlVirtRole, 9

Device Functions, 17

- mtmlDeviceFreeGpu, 18
- mtmlDeviceFreeMemory, 18
- mtmlDeviceFreeVpu, 19
- mtmlDeviceGetBrand, 19
- mtmlDeviceGetGpuPath, 20
- mtmlDeviceGetIndex, 20
- mtmlDeviceGetName, 21
- mtmlDeviceGetPciInfo, 21
- mtmlDeviceGetPowerUsage, 22
- mtmlDeviceGetPrimaryPath, 22
- mtmlDeviceGetProperty, 23
- mtmlDeviceGetRenderPath, 23
- mtmlDeviceGetUUID, 24

- mtmlDeviceGetVbiosVersion, 24

- mtmlDeviceInitGpu, 25

- mtmlDeviceInitMemory, 26

- mtmlDeviceInitVpu, 26

Device Virtualization Functions, 27

- mtmlDeviceCountActiveVirtDevices, 27

- mtmlDeviceCountAvailVirtDevices, 28

- mtmlDeviceCountAvailVirtTypes, 28

- mtmlDeviceCountSupportedVirtTypes, 29

- mtmlDeviceFreeVirtDevice, 29

- mtmlDeviceGetActiveVirtDeviceUuids, 30

- mtmlDeviceGetAvailVirtTypes, 31

- mtmlDeviceGetPhyDeviceUuid, 31

- mtmlDeviceGetSupportedVirtTypes, 32

- mtmlDeviceGetVirtType, 32

- mtmlDeviceInitVirtDevice, 33

Functional Structure Definitions, 10

GPU Functions, 34

- mtmlGpuGetClock, 34

- mtmlGpuGetMaxClock, 34

- mtmlGpuGetTemperature, 35

- mtmlGpuGetUtilization, 35

Library Functions, 11

- mtmlLibraryCountDevice, 11

- mtmlLibraryFreeDevice, 12

- mtmlLibraryFreeSystem, 12

- mtmlLibraryGetVersion, 13

- mtmlLibraryInit, 13

- mtmlLibraryInitDeviceByIndex, 14

- mtmlLibraryInitDeviceByUuid, 15

- mtmlLibraryInitSystem, 15

- mtmlLibraryShutDown, 16

Memory Functions, 36

- mtmlMemoryGetClock, 36

- mtmlMemoryGetMaxClock, 37

- mtmlMemoryGetTotal, 37

- mtmlMemoryGetUsed, 38

- mtmlMemoryGetUtilization, 38

MTML_BRAND_COUNT

- Constant definitions, 9

MTML_BRAND_MTT

- Constant definitions, 9

MTML_BRAND_UNKNOWN

- Constant definitions, 9

MTML_DEVICE_NAME_BUFFER_SIZE

- Constant definitions, 7

MTML_DEVICE_PATH_BUFFER_SIZE	mtmlDeviceCountAvailVirtDevices
Constant definitions, 7	Device Virtualization Functions, 28
MTML_DEVICE_PCI_BUS_ID_FMT	mtmlDeviceCountAvailVirtTypes
Constant definitions, 7	Device Virtualization Functions, 28
MTML_DEVICE_PCI_SDF_BUFFER_SIZE	mtmlDeviceCountSupportedVirtTypes
Constant definitions, 7	Device Virtualization Functions, 29
MTML_DEVICE_UUID_BUFFER_SIZE	mtmlDeviceFreeGpu
Constant definitions, 7	Device Functions, 18
MTML_DEVICE_VBIOS_VERSION_BUFFER_SIZE	mtmlDeviceFreeMemory
Constant definitions, 7	Device Functions, 18
MTML_DRIVER_VERSION_BUFFER_SIZE	mtmlDeviceFreeVirtDevice
Constant definitions, 7	Device Virtualization Functions, 29
MTML_ERROR_DRIVER_FAILURE	mtmlDeviceFreeVpu
Constant definitions, 9	Device Functions, 19
MTML_ERROR_DRIVER_NOT_LOADED	mtmlDeviceGetActiveVirtDeviceUuids
Constant definitions, 9	Device Virtualization Functions, 30
MTML_ERROR_INSUFFICIENT_MEMORY	mtmlDeviceGetAvailVirtTypes
Constant definitions, 9	Device Virtualization Functions, 31
MTML_ERROR_INSUFFICIENT_SIZE	mtmlDeviceGetBrand
Constant definitions, 9	Device Functions, 19
MTML_ERROR_INVALID_ARGUMENT	mtmlDeviceGetGpuPath
Constant definitions, 9	Device Functions, 20
MTML_ERROR_NO_PERMISSION	mtmlDeviceGetIndex
Constant definitions, 9	Device Functions, 20
MTML_ERROR_NOT_FOUND	mtmlDeviceGetName
Constant definitions, 9	Device Functions, 21
MTML_ERROR_NOT_SUPPORTED	mtmlDeviceGetPciInfo
Constant definitions, 9	Device Functions, 21
MTML_ERROR_UNKNOWN	mtmlDeviceGetPhyDeviceUuid
Constant definitions, 9	Device Virtualization Functions, 31
MTML_LIBRARY_VERSION_BUFFER_SIZE	mtmlDeviceGetPowerUsage
Constant definitions, 8	Device Functions, 22
MTML_SUCCESS	mtmlDeviceGetPrimaryPath
Constant definitions, 9	Device Functions, 22
MTML_VIRT_ROLE_COUNT	mtmlDeviceGetProperty
Constant definitions, 10	Device Functions, 23
MTML_VIRT_ROLE_HOST_VIRTDEVICE	mtmlDeviceGetRenderPath
Constant definitions, 9	Device Functions, 23
MTML_VIRT_ROLE_NONE	mtmlDeviceGetSupportedVirtTypes
Constant definitions, 9	Device Virtualization Functions, 32
MTML_VIRT_TYPE_API_BUFFER_SIZE	mtmlDeviceGetUUID
Constant definitions, 8	Device Functions, 24
MTML_VIRT_TYPE_CLASS_BUFFER_SIZE	mtmlDeviceGetVbiosVersion
Constant definitions, 8	Device Functions, 24
MTML_VIRT_TYPE_ID_BUFFER_SIZE	mtmlDeviceGetVirtType
Constant definitions, 8	Device Virtualization Functions, 32
MTML_VIRT_TYPE_NAME_BUFFER_SIZE	mtmlDeviceInitGpu
Constant definitions, 8	Device Functions, 25
MtmlBrandType	mtmlDeviceInitMemory
Constant definitions, 8	Device Functions, 26
MtmlCodecSessionMetrics, 46	mtmlDeviceInitVirtDevice
MtmlCodecType	Device Virtualization Functions, 33
Constant definitions, 9	mtmlDeviceInitVpu
MtmlCodecUtil, 46	Device Functions, 26
MtmlDevice	MtmlDeviceProperty, 47
Opaque Data Structure Definitions, 10	virtCap, 48
mtmlDeviceCountActiveVirtDevices	virtRole, 48
Device Virtualization Functions, 27	MtmlGpu

- Opaque Data Structure Definitions, [10](#)
- mtmlGpuGetClock
 - GPU Functions, [34](#)
- mtmlGpuGetMaxClock
 - GPU Functions, [34](#)
- mtmlGpuGetTemperature
 - GPU Functions, [35](#)
- mtmlGpuGetUtilization
 - GPU Functions, [35](#)
- MtmlLibrary
 - Opaque Data Structure Definitions, [10](#)
- mtmlLibraryCountDevice
 - Library Functions, [11](#)
- mtmlLibraryFreeDevice
 - Library Functions, [12](#)
- mtmlLibraryFreeSystem
 - Library Functions, [12](#)
- mtmlLibraryGetVersion
 - Library Functions, [13](#)
- mtmlLibraryInit
 - Library Functions, [13](#)
- mtmlLibraryInitDeviceByIndex
 - Library Functions, [14](#)
- mtmlLibraryInitDeviceByUuid
 - Library Functions, [15](#)
- mtmlLibraryInitSystem
 - Library Functions, [15](#)
- mtmlLibraryShutDown
 - Library Functions, [16](#)
- MtmlMemory
 - Opaque Data Structure Definitions, [11](#)
- mtmlMemoryGetClock
 - Memory Functions, [36](#)
- mtmlMemoryGetMaxClock
 - Memory Functions, [37](#)
- mtmlMemoryGetTotal
 - Memory Functions, [37](#)
- mtmlMemoryGetUsed
 - Memory Functions, [38](#)
- mtmlMemoryGetUtilization
 - Memory Functions, [38](#)
- MtmlPciInfo, [45](#)
- MtmlReturn
 - Constant definitions, [9](#)
- MtmlSystem
 - Opaque Data Structure Definitions, [11](#)
- mtmlSystemGetDriverVersion
 - System Functions, [17](#)
- MtmlVirtRole
 - Constant definitions, [9](#)
- MtmlVirtType, [47](#)
- MtmlVpu
 - Opaque Data Structure Definitions, [11](#)
- mtmlVpuGetClock
 - VPU Functions, [39](#)
- mtmlVpuGetCodecCapacity
 - VPU Functions, [40](#)
- mtmlVpuGetDecoderSessionMetrics
 - VPU Functions, [40](#)
- mtmlVpuGetDecoderSessionStates
 - VPU Functions, [41](#)
- mtmlVpuGetEncoderSessionMetrics
 - VPU Functions, [41](#)
- mtmlVpuGetEncoderSessionStates
 - VPU Functions, [42](#)
- mtmlVpuGetMaxClock
 - VPU Functions, [43](#)
- mtmlVpuGetUtilization
 - VPU Functions, [43](#)
- Opaque Data Structure Definitions, [10](#)
 - MtmlDevice, [10](#)
 - MtmlGpu, [10](#)
 - MtmlLibrary, [10](#)
 - MtmlMemory, [11](#)
 - MtmlSystem, [11](#)
 - MtmlVpu, [11](#)
- System Functions, [17](#)
 - mtmlSystemGetDriverVersion, [17](#)
- virtCap
 - MtmlDeviceProperty, [48](#)
- virtRole
 - MtmlDeviceProperty, [48](#)
- VPU Functions, [39](#)
 - mtmlVpuGetClock, [39](#)
 - mtmlVpuGetCodecCapacity, [40](#)
 - mtmlVpuGetDecoderSessionMetrics, [40](#)
 - mtmlVpuGetDecoderSessionStates, [41](#)
 - mtmlVpuGetEncoderSessionMetrics, [41](#)
 - mtmlVpuGetEncoderSessionStates, [42](#)
 - mtmlVpuGetMaxClock, [43](#)
 - mtmlVpuGetUtilization, [43](#)