MODULE *Nv0000Classes*



EXTENDS

*Naturals*,
*Sequences*,

Internal Modules.

*RsAccess*

CONSTANTS

| | |
|---|---|
| *Nv0000CtrlReserved*, | Reserved. |
| *Nv0000CtrlSystem*, | Command for system. |
| *Nv0000CtrlGpu*, | Command for *GPU*. |
| *Nv0000CtrlGsync*, | Command for *GSYNC*. |
| *Nv0000CtrlDiag*, | Command for Diagnostics. |
| *Nv0000CtrlEvent*, | Command for Event interactions. |
| *Nv0000CtrlNvd*, | Command for *NVIDIA* Debug Dumps. |
| *Nv0000CtrlSwInstr*, | Command for *SW* Instruction. |
| *Nv0000CtrlGspC*, | Unknown. |
| *Nv0000CtrlProc*, | Command for /proc commands. |
| *Nv0000CtrlSyncGpuBoost*, | Command for *SLI GPU* Boosts. |
| *Nv0000CtrlGpuAcct*, | Command for *GPU* accounting. |
| *Nv0000CtrlVgpu*, | Command for *vGPUs*. |
| *Nv0000CtrlClient* | Command for Clients. |

## CLIENTS

CONSTANTS

| | |
|---|---|
| *Nv0000CtrlCmdGetAddrSpaceType*, | Query memory address space type associated with an object. |
| *Nv0000CtrlCmdGetHandleInfo*, | Query information on a handle. |
| *Nv0000CtrlCmdClientGetAccessRights*, | Gets access rights for an object, the object does not need to be owned by the client requesting the access rights. |
| *Nv0000CtrlCmdSetInheritedShare*, | Sets an inherited policy list. |

$Nv0000CtrlCmdGetChildHandle$, Gets the child handle of a given type.

$Nv0000CtrlCmdShareObject$ Deprecating this command, but it is designed to share an object.

Possible address space types.

CONSTANTS

$AddrSpaceTypeInvalid$,
$AddrSpaceTypeSysmem$,
$AddrSpaceTypeVidmem$,
$AddrSpaceTypeRegmem$,
$AddrSpaceTypeFabric$

$AddrSpaceTypes \triangleq$
{$AddrSpaceTypeInvalid$
, $AddrSpaceTypeSysmem$
, $AddrSpaceTypeVidmem$
, $AddrSpaceTypeRegmem$
, $AddrSpaceTypeFabric$
}

Object for getting the address space type.

$NvGetAddrSpaceType \triangleq$
[$object : Nat$ Object handle we attempt to look up. [IN ]

, $mapFlags : Nat$ Flags for mapping a space addr type. [IN ]

, $addrSpaceType : AddrSpaceTypes$ Type of address space. [OUT]
]

Possible handle info lookups.

CONSTANTS

$GetHandleInfoInvalid$, Invalid lookup.
$GetHandleInfoParent$, Parent device handle.
$GetHandleInfoClassId$ Class *Id* of the device.

$HandleInfoLookup \triangleq$
{$GetHandleInfoInvalid$
, $GetHandleInfoParent$
, $GetHandleInfoClassId$
}

Object for getting the handle information.

$NvGetHandleInfo \triangleq$
[$object : Nat$ Object to look up. [IN ]
, $index : HandleInfoLookup$ Handle info lookup type. [IN ]

, *result* : *Nat* — Result. [OUT]
]

Object to get the access rights for an object.
*NvGetAccessRights* $\triangleq$
[*object* : *Nat* — Object to look up. [IN ]
, *client* : *Nat* — Client that owns the object. [IN ]
, *result* : *AccessMask* — Result of the lookup. [OUT]
]

Object to set inherited share policy.
*NvSetInheritedSharePolicy* $\triangleq$ *RsSharePolicy*

Object to get the child handle.
*NvGetChildHandle* $\triangleq$
[*parent* : *Nat* — Parent object handle. [IN ]
, *classid* : *Nat* — Class id of the child. [IN ]
, *object* : *Nat* — Object *ID*. [OUT]
]

Object to share another object.
NOTE: Avoid for releases after *R*450.
*NvShareObject* $\triangleq$
[*object* : *Nat* — Object to share. [IN ]
, *share* : *RsSharePolicy* — Sharing policy. [IN ]
]

## DIAGNOSTICS

CONSTANTS
*Nv*0000*CtrlCmdGetLockMeter*, — Returns the current lock meter.
*Nv*0000*CtrlCmdSetLockMeter*, — Sets the current lock meter.
*Nv*0000*CtrlCmdGetLockMeterEntries*, — Gets a list of lock meter entries.
*Nv*0000*CtrlCmdProfileRpc*, — Profiles an *RPC* in *VGX* mode.
*Nv*0000*CtrlCmdDumpRpc* — Dumps *RPC* runtime information.

Possible lock meter states.
CONSTANTS
*LockMeterDisabled*, — Disables lock metering.
*LockMeterEnabled*, — Enables lock metering.
*LockMeterReset* — Clears the locks, but requires it is disabled first.

*LockMeterStates* $\triangleq$
{*LockMeterDisabled*
, *LockMeterEnabled*

, *LockMeterReset*
}

Object for getting a meter lock state.

*NvGetLockMeter* $\triangleq$

[*state* : {*LockMeterDisabled*, *LockMeterEnabled*} — Whether the lock meter is enabled or disabled.

, *count* : *Nat* — Number of entries available.

, *missedCount* : *Nat* — Number of missed entries.

, *circularBuffer* : BOOLEAN — If the buffer is circular or sequential.

]

Object for setting a meter lock state.

*NvSetMeterLock* $\triangleq$

[*state* : *LockMeterStates* — Possible lock meter states.

, *circularBuffer* : BOOLEAN — If the buffer is circular or sequential.

]

Possible metering tags.

CONSTANTS

*LockMeterTagAcquireSema*,
*LockMeterTagAcquireSemaForced*,
*LockMeterTagAcquireSemaCond*,
*LockMeterTagReleaseSema*,
*LockMeterTagAcquireApi*,
*LockMeterTagReleaseApi*,
*LockMeterTagAcquireGpus*,
*LockMeterTagReleaseGpus*,
*LockMeterTagData*,
*LockMeterTagRmCtrl*,
*LockMeterTagCfgGet*,
*LockMeterTagCfgSet*,
*LockMeterTagCfgGetEx*,
*LockMeterTagCfgSetEx*,
*LockMeterTagVidHeap*,
*LockMeterTagMapMem*,
*LockMeterTagUnMapMem*,
*LockMeterTagMapMemDma*,
*LockMeterTagUnMapMemDma*,
*LockMeterTagAlloc*,
*LockMeterTagAllocMem*,
*LockMeterTagDupObject*,
*LockMeterTagFreeClient*,

$LockMeterTagFreeDevice$,
$LockMeterTagFreeSubDevice$,
$LockMeterTagFreeSubDeviceDiag$,
$LockMeterTagFreeDisp$,
$LockMeterTagFreeDispCmn$,
$LockMeterTagFreeChannel$,
$LockMeterTagFreeChannelMpeg$,
$LockMeterTagFreeChannelDisp$,
$LockMeterTagIdleChannels$,
$LockMeterTagBindCtxDma$,
$LockMeterTagAllocCtxDma$,
$LockMeterTagIsr$,
$LockMeterTagDpc$

$LockMeterTags \triangleq$
{$LockMeterTagAcquireSema$, $LockMeterTagAcquireSemaForced$,
$LockMeterTagAcquireSemaCond$, $LockMeterTagReleaseSema$,
$LockMeterTagAcquireApi$, $LockMeterTagReleaseApi$,
$LockMeterTagAcquireGpus$, $LockMeterTagReleaseGpus$,
$LockMeterTagData$, $LockMeterTagRmCtrl$,
$LockMeterTagCfgGet$, $LockMeterTagCfgSet$,
$LockMeterTagCfgGetEx$, $LockMeterTagCfgSetEx$,
$LockMeterTagVidHeap$, $LockMeterTagMapMem$,
$LockMeterTagUnMapMem$, $LockMeterTagMapMemDma$,
$LockMeterTagUnMapMemDma$, $LockMeterTagAlloc$,
$LockMeterTagAllocMem$, $LockMeterTagDupObject$,
$LockMeterTagFreeClient$, $LockMeterTagFreeDevice$,
$LockMeterTagFreeSubDevice$, $LockMeterTagFreeSubDeviceDiag$,
$LockMeterTagFreeDisp$, $LockMeterTagFreeDispCmn$,
$LockMeterTagFreeChannel$, $LockMeterTagFreeChannelMpeg$,
$LockMeterTagFreeChannelDisp$, $LockMeterTagIdleChannels$,
$LockMeterTagBindCtxDma$, $LockMeterTagAllocCtxDma$,
$LockMeterTagIsr$, $LockMeterTagDpc$}

Lock metering entry.
NOTE: We are ignoring the following for this portion:
- freq
- line
- filename
- *cpuNum*
- irql
- *threadId*

$NvLockMeterEntry \triangleq$
[$counter : Nat$ Nanoseconds since last boot.
, $tag : LockMeterTags$ Which kind of tag is this meter.

, $data0 : Nat$ Tag specific information.
, $data1 : Nat$
, $data2 : Nat$
]

Object for getting the lock meter entries.
$NvGetLockMeterEntries \triangleq Seq(NvLockMeterEntry)$

Profiles the *RPC* information.
NOTE: Ignoring both start and end time both in nanoseconds.
NOTE: Available *RPCs* are not known yet.
NOTE: Extra data is not known yet.
$NvProfileRpcEntry \triangleq$
[$rpcDataTag : \{\}$ Data tag for the *RPC* command.
, $rpcExtraData : \{\}$ Extra data related to the *RPC*.
]

$NvProfileRpcEntries \triangleq Seq(NvProfileRpcEntry)$

Constants for profiling an *RPC* command.
CONSTANTS
$ProfileRpcCmdDisable$, Disables the *RPC* profiling.
$ProfileRpcCmdEnable$, Enables the *RPC* profiling.
$ProfileRpcCmdReset$ Resets the *RPC* profiling.

Profile the requested rpc command.
$NvProfileRpcCmd \triangleq$
{$ProfileRpcCmdDisable$
, $ProfileRpcCmdEnable$
, $ProfileRpcCmdReset$
}

Object to dump the *RPC* runtime information.
NOTE: This only works running in *VGX* mode.
NOTE: Ignoring *elapsedTimeInNs*. [OUT]
$NvDumpRpc \triangleq$
[$firstEntryOffset : Nat$ Offset for first entry. [IN ]
, $outputEntryCount : Nat$ Count of entries in profiler buffer. [OUT]
, $remainingEntryCount : Nat$ Remaining number of entries. [OUT]
, $profilers : NvProfileRpcEntries$ Profiler entries. [OUT]
]

EVENTS

CONSTANTS

$Nv0000CtrlCmdSetNotification$, Sets an event notification for system events.

$Nv0000CtrlCmdGetSystemEventStatus$ Returns the status of a specified system event type.

Notification actions.

CONSTANTS

$EventSetNotificationActionDisable$, Disables event notification.
$EventSetNotificationActionSingle$, Enables for a single shot event.
$EventSetNotificationActionRepeat$ Repeats the event notification.

$$EventSetNotifications \triangleq$$
$$\{EventSetNotificationActionDisable$$
$$, EventSetNotificationActionSingle$$
$$, EventSetNotificationActionRepeat$$
$$\}$$

Event types.

CONSTANTS

$NotificationDisplayChange$, Notification if a display changes.

$NotificationEventNonePending$, Notification if none are pending.

$NotificationVmStart$, *VM* started notification.
$NotificationGpuBindEvent$, *VM* additional gpu bound.
$NotificationTelemetryReport$ Report from the telemetry.

$$NvNotificationEvents \triangleq$$
$$\{NotificationDisplayChange$$
$$, NotificationEventNonePending$$
$$, NotificationVmStart$$
$$, NotificationGpuBindEvent$$
$$, NotificationTelemetryReport$$
$$\}$$

Object to set an event notification.
*TODO*: Get a list of event classes.

$$NvEventSetNotification \triangleq$$
$$[event : NvNotificationEvents$$ In the set of event classes.
$$, action : EventSetNotifications$$ Action to preform.
$$]$$

Object to get the system event status.

$$NvGetSystemEventStatus \triangleq$$
$$[event : NvNotificationEvents$$ Event to get the notifier on.
$$, status : Nat$$ *RmStatus*.
$$]$$

*GPU*

This is the $NV0000$ Allocation Object.

$Nv0000AllocParams \triangleq [client : Nat$ — This is the client id which allows the process to interact with the driver.

$, processID : Nat$ — This is the *processID* to lock the resources to. There is also a name parameter which we removed as it serves no purpose in our specs.

$]$

$$
\begin{aligned}
&Nv0000Ctrls \triangleq \\
&\quad [Nv0000CtrlReserved : \{\}] \cup \\
&\quad [Nv0000CtrlClient : \{Nv0000CtrlCmdGetAddrSpaceType \\
&\qquad\qquad , Nv0000CtrlCmdGetHandleInfo \\
&\qquad\qquad , Nv0000CtrlCmdClientGetAccessRights \\
&\qquad\qquad , Nv0000CtrlCmdSetInheritedShare \\
&\qquad\qquad , Nv0000CtrlCmdGetChildHandle \\
&\qquad\qquad , Nv0000CtrlCmdShareObject \\
&\qquad\qquad \}] \cup \\
&\quad [Nv0000CtrlEvent : \{Nv0000CtrlCmdSetNotification \\
&\qquad\qquad , Nv0000CtrlCmdGetSystemEventStatus \\
&\qquad\qquad \}] \cup \\
&\quad [Nv0000CtrlDiag \ : \{Nv0000CtrlCmdGetLockMeter \\
&\qquad\qquad , Nv0000CtrlCmdSetLockMeter \\
&\qquad\qquad , Nv0000CtrlCmdGetLockMeterEntries \\
&\qquad\qquad , Nv0000CtrlCmdProfileRpc \\
&\qquad\qquad , Nv0000CtrlCmdDumpRpc \\
&\qquad\qquad \}] \cup \\
&\quad [Nv0000CtrlGspC : \{\}]
\end{aligned}
$$

* Modification History
* Last modified *Tue Jun* 21 15:39:35 *EDT* 2022 by *mbuchel*
* Created *Fri Jun* 17 11:04:19 *EDT* 2022 by *mbuchel*