



DATACENTER GPU MANAGER API MANUAL

v1.5.6 | November 2018

Reference Manual



TABLE OF CONTENTS

Chapter 1. Change Log.....	1
Chapter 2. Modules.....	2
2.1. Administrative.....	3
Init and Shutdown.....	3
2.1.1. Init and Shutdown.....	3
dcgmlInit.....	3
dcgmShutdown.....	3
dcgmStartEmbedded.....	4
dcgmStopEmbedded.....	4
dcgmConnect.....	5
dcgmConnect_v2.....	5
dcgmDisconnect.....	6
2.2. System.....	6
Discovery.....	7
Grouping.....	7
Field Grouping.....	7
Status handling.....	7
2.2.1. Discovery.....	7
dcgmGetAllDevices.....	7
dcgmGetAllSupportedDevices.....	8
dcgmGetDeviceAttributes.....	8
dcgmGetEntityGroupEntities.....	9
dcgmGetNvLinkLinkStatus.....	10
2.2.2. Grouping.....	10
dcgmGroupCreate.....	10
dcgmGroupDestroy.....	11
dcgmGroupAddDevice.....	12
dcgmGroupAddEntity.....	12
dcgmGroupRemoveDevice.....	13
dcgmGroupRemoveEntity.....	13
dcgmGroupGetInfo.....	14
dcgmGroupGetAllIds.....	15
2.2.3. Field Grouping.....	15
dcgmFieldGroupCreate.....	15
dcgmFieldGroupDestroy.....	16
dcgmFieldGroupGetInfo.....	17
dcgmFieldGroup GetAll.....	17
2.2.4. Status handling.....	18
dcgmStatusCreate.....	18
dcgmStatusDestroy.....	18

dcgmStatusGetCount.....	19
dcgmStatusPopError.....	19
dcgmStatusClear.....	20
2.3. Configuration.....	20
Setup and management.....	20
Manual Invocation.....	20
2.3.1. Setup and management.....	20
dcgmConfigSet.....	21
dcgmConfigGet.....	22
2.3.2. Manual Invocation.....	23
dcgmConfigEnforce.....	23
2.4. Field APIs.....	24
dcgmWatchFields.....	24
dcgmUnwatchFields.....	25
dcgmGetValuesSince.....	25
dcgmGetValuesSince_v2.....	26
dcgmGetLatestValues.....	27
dcgmGetLatestValues_v2.....	28
dcgmGetLatestValuesForFields.....	29
dcgmEntityGetLatestValues.....	29
dcgmEntitiesGetLatestValues.....	30
2.5. Process Statistics.....	31
dcgmWatchPidFields.....	31
dcgmGetPidInfo.....	32
2.6. Job Statistics.....	33
dcgmWatchJobFields.....	33
dcgmJobStartStats.....	34
dcgmJobStopStats.....	34
dcgmJobGetStats.....	35
dcgmJobRemove.....	35
dcgmJobRemoveAll.....	36
2.7. Health Monitor.....	36
dcgmHealthSet.....	37
dcgmHealthGet.....	37
dcgmHealthCheck.....	38
2.8. Policies.....	39
Setup and Management.....	39
Manual Invocation.....	39
2.8.1. Setup and Management.....	39
dcgmPolicySet.....	39
dcgmPolicyGet.....	40
dcgmPolicyRegister.....	41
dcgmPolicyUnregister.....	42

2.8.2. Manual Invocation.....	42
dcgmActionValidate.....	43
dcgmActionValidate_v2.....	43
dcgmRunDiagnostic.....	44
2.9. Topology.....	45
dcgmGetDeviceTopology.....	45
dcgmGetGroupTopology.....	46
2.10. Metadata.....	46
dcgmlntrospectToggleState.....	46
dcgmlntrospectGetFieldsMemoryUsage.....	47
dcgmlntrospectGetHostengineMemoryUsage.....	48
dcgmlntrospectGetFieldsExecTime.....	48
dcgmlntrospectUpdateAll.....	49
2.11. Topology.....	50
dcgmSelectGpusByTopology.....	50
dcgmGetFieldSummary.....	51
2.12. Modules.....	51
dcgmModuleBlacklist.....	51
dcgmModuleGetStatuses.....	52
2.13. Enums and Macros.....	52
dcgmOperationMode_t.....	52
dcgmOrder_t.....	53
dcgmReturn_t.....	53
dcgmGroupType_t.....	55
dcgmConfigType_t.....	55
dcgmConfigPowerLimitType_t.....	55
DCGM_INT32_BLANK.....	55
DCGM_INT64_BLANK.....	56
DCGM_FP64_BLANK.....	56
DCGM_STR_BLANK.....	56
DCGM_INT32_NOT_FOUND.....	56
DCGM_INT64_NOT_FOUND.....	56
DCGM_FP64_NOT_FOUND.....	56
DCGM_STR_NOT_FOUND.....	56
DCGM_INT32_NOT_SUPPORTED.....	56
DCGM_INT64_NOT_SUPPORTED.....	56
DCGM_FP64_NOT_SUPPORTED.....	57
DCGM_STR_NOT_SUPPORTED.....	57
DCGM_INT32_NOT_PERMISSIONED.....	57
DCGM_INT64_NOT_PERMISSIONED.....	57
DCGM_FP64_NOT_PERMISSIONED.....	57
DCGM_STR_NOT_PERMISSIONED.....	57
DCGM_INT32_IS_BLANK.....	57

DCGM_INT64_IS_BLANK.....	58
DCGM_FP64_IS_BLANK.....	58
DCGM_STR_IS_BLANK.....	58
DCGM_MAX_NUM_DEVICES.....	58
DCGM_NVLINK_MAX_LINKS_PER_GPU.....	58
DCGM_MAX_NUM_SWITCHES.....	58
DCGM_NVLINK_MAX_LINKS_PER_NVSWITCH.....	58
DCGM_MAX_VGPU_INSTANCES_PER_PGPU.....	58
DCGM_MAX_NUM_VGPU_DEVICES.....	58
DCGM_MAX_STR_LENGTH.....	59
DCGM_MAX_CLOCKS.....	59
DCGM_MAX_NUM_GROUPS.....	59
DCGM_MAX_FBC_SESSIONS.....	59
DCGM_VGPU_NAME_BUFFER_SIZE.....	59
DCGM_GRID_LICENSE_BUFFER_SIZE.....	59
DCGM_CONFIG_COMPUTEMODE_DEFAULT.....	59
DCGM_CONFIG_COMPUTEMODE_PROHIBITED.....	59
DCGM_CONFIG_COMPUTEMODE_EXCLUSIVE_PROCESS.....	59
DCGM_HE_PORT_NUMBER.....	59
MAKE_DCGM_VERSION.....	60
DCGM_GROUP_ALL_GPUS.....	60
DCGM_GROUP_MAX_ENTITIES.....	60
2.14. Structure definitions.....	60
dcgmConnectV2Params_v1.....	61
dcgmConnectV2Params_v2.....	61
dcgmGroupInfo_v1.....	61
dcgmGroupEntityPair_t.....	61
dcgmGroupInfo_v2.....	61
dcgmFieldGroupInfo_v1.....	61
dcgmErrorInfo_t.....	61
dcgmClockSet_v1.....	61
dcgmDeviceSupportedClockSets_v1.....	61
dcgmDevicePidAccountingStats_v1.....	61
dcgmDeviceThermals_v1.....	61
dcgmDevicePowerLimits_v1.....	61
dcgmDeviceIdentifiers_v1.....	61
dcgmDeviceMemoryUsage_v1.....	61
dcgmDeviceVgpuUtilInfo_v1.....	61
dcgmDeviceEncStats_v1.....	61
dcgmDeviceFbcStats_v1.....	61
dcgmDeviceFbcSessionInfo_v1.....	61
dcgmDeviceFbcSessions_v1.....	62
dcgmDeviceVgpuEncSessions_v1.....	62

dcgmDeviceVgpuProcessUtilInfo_v1.....	62
dcgmDeviceVgpuids_v1.....	62
dcgmDeviceVgpuTypeInfo_v1.....	62
dcgmDeviceAttributes_v1.....	62
dcgmVgpuDeviceAttributes_v5.....	62
dcgmVgpuInstanceAttributes_v1.....	62
dcgmConfigPerfStateSettings_t.....	62
dcgmConfigPowerLimit_t.....	62
dcgmConfig_v1.....	62
dcgmVgpuConfig_v1.....	62
dcgmPolicyConditionParms_t.....	62
dcgmPolicyViolationNotify_t.....	62
dcgmPolicy_v1.....	62
dcgmPolicyConditionDbe_t.....	62
dcgmPolicyConditionPci_t.....	62
dcgmPolicyConditionMpr_t.....	62
dcgmPolicyConditionThermal_t.....	63
dcgmPolicyConditionPower_t.....	63
dcgmPolicyConditionNvlink_t.....	63
dcgmPolicyConditionXID_t.....	63
dcgmPolicyCallbackResponse_v1.....	63
dcgmFieldValue_v1.....	63
dcgmFieldValue_v2.....	63
dcgmStatSummaryInt64_t.....	63
dcgmStatSummaryInt32_t.....	63
dcgmStatSummaryFp64_t.....	63
dcgmHealthResponse_v1.....	63
dcgmHealthResponse_v2.....	63
dcgmProcessUtilInfo_t.....	63
dcgmProcessUtilSample_t.....	63
dcgmPidSingleInfo_t.....	63
dcgmPidInfo_v1.....	63
dcgmGpuUsageInfo_t.....	63
dcgmJobInfo_v2.....	63
dcgmRunningProcess_v1.....	64
dcgmDiagResponsePerGpu_t.....	64
dcgmDiagResponse_v3.....	64
dcgmDeviceTopology_v1.....	64
dcgmGroupTopology_v1.....	64
dcgmIntrospectContext_v1.....	64
dcgmIntrospectFieldsExecTime_v1.....	64
dcgmIntrospectFullFieldsExecTime_v1.....	64
dcgmIntrospectMemory_v1.....	64

dcgmIntrospectFullMemory_v1.....	64
dcgmIntrospectCpuUtil_v1.....	64
dcgmNvLinkGpuLinkStatus_t.....	64
dcgmNvLinkNvSwitchLinkStatus_t.....	64
dcgmNvLinkStatus_v1.....	64
dcgmModuleGetStatusesModule_t.....	64
dcgmPolicyCondition_t.....	64
dcgmPolicyMode_t.....	65
dcgmPolicyIsolation_t.....	65
dcgmPolicyAction_t.....	65
dcgmPolicyValidation_t.....	66
dcgmPolicyFailureResp_t.....	66
dcgmHealthSystems_t.....	66
dcgmHealthWatchResults_t.....	67
dcgmDiagnosticLevel_t.....	67
dcgmDiagResult_t.....	68
dcgmPerGpuTestIndices_t.....	68
dcgmGpuTopologyLevel_t.....	68
dcgmIntrospectLevel_t.....	69
dcgmIntrospectState_t.....	70
dcgmGpuNVLinkErrorType_t.....	70
dcgmNvLinkLinkState_t.....	70
dcgmModuleId_t.....	70
dcgmModuleStatus_t.....	71
dcgmHandle_t.....	71
dcgmGpuGrp_t.....	71
dcgmFieldGrp_t.....	71
dcgmStatus_t.....	72
dcgmConnectV2Params_t.....	72
dcgmGroupInfo_t.....	72
dcgmClockSet_t.....	72
dcgmDeviceSupportedClockSets_t.....	72
dcgmDevicePidAccountingStats_t.....	72
dcgmDeviceThermals_t.....	72
dcgmDevicePowerLimits_t.....	72
dcgmDeviceIdentifiers_t.....	72
dcgmDeviceMemoryUsage_t.....	72
dcgmDeviceVgpuUtilInfo_t.....	72
dcgmDeviceEncStats_t.....	73
dcgmDeviceFbcStats_t.....	73
dcgmDeviceFbcSessionInfo_t.....	73
dcgmDeviceFbcSessions_t.....	73
dcgmDeviceVgpuEncSessions_t.....	73

dcgmDeviceVgpuProcessUtilInfo_t.....	73
dcgmDeviceVgpuids_t.....	73
dcgmDeviceVgpuTypeInfo_t.....	73
dcgmDeviceAttributes_t.....	73
dcgmVgpuDeviceAttributes_t.....	73
dcgmVgpuInstanceAttributes_t.....	73
dcgmConfig_t.....	74
dcgmVgpuConfig_t.....	74
fpRecvUpdates.....	74
dcgmPolicy_t.....	74
dcgmPolicyCallbackResponse_t.....	74
dcgmFieldValue_t.....	74
dcgmFieldValueEnumeration_f.....	74
dcgmFieldValueEntityEnumeration_f.....	74
dcgmHealthResponse_t.....	75
dcgmPidInfo_t.....	75
dcgmJobInfo_t.....	75
dcgmRunningProcess_t.....	75
dcgmDiagResponse_t.....	75
dcgmDeviceTopology_t.....	75
dcgmGroupTopology_t.....	75
dcgmlntrospectContext_t.....	75
dcgmlntrospectFieldsExecTime_t.....	75
dcgmlntrospectFullFieldsExecTime_t.....	76
dcgmlntrospectMemory_t.....	76
dcgmlntrospectFullMemory_t.....	76
dcgmlntrospectCpuUtil_t.....	76
dcgmRunDiag_t.....	76
dcgmConnectV2Params_version1.....	76
dcgmConnectV2Params_version2.....	76
dcgmConnectV2Params_version.....	76
dcgmGroupInfo_version1.....	76
dcgmGroupInfo_version2.....	77
dcgmGroupInfo_version.....	77
DCGM_MAX_NUM_FIELD_GROUPS.....	77
DCGM_MAX_FIELD_IDS_PER_FIELD_GROUP.....	77
dcgmFieldGroupInfo_version1.....	77
dcgmFieldGroupInfo_version.....	77
dcgmAllFieldGroup_version1.....	77
dcgmAllFieldGroup_version.....	77
dcgmClockSet_version1.....	77
dcgmClockSet_version.....	78
dcgmDeviceSupportedClockSets_version1.....	78

dcgmDeviceSupportedClockSets_version.....	78
dcgmDevicePidAccountingStats_version1.....	78
dcgmDevicePidAccountingStats_version.....	78
dcgmDeviceThermals_version1.....	78
dcgmDeviceThermals_version.....	78
dcgmDevicePowerLimits_version1.....	78
dcgmDevicePowerLimits_version.....	79
dcgmDeviceIdentifiers_version1.....	79
dcgmDeviceIdentifiers_version.....	79
dcgmDeviceMemoryUsage_version1.....	79
dcgmDeviceMemoryUsage_version.....	79
dcgmDeviceVgpuUtilInfo_version1.....	79
dcgmDeviceVgpuUtilInfo_version.....	79
dcgmDeviceEncStats_version1.....	79
dcgmDeviceEncStats_version.....	80
dcgmDeviceFbcStats_version1.....	80
dcgmDeviceFbcStats_version.....	80
dcgmDeviceFbcSessionInfo_version1.....	80
dcgmDeviceFbcSessionInfo_version.....	80
dcgmDeviceFbcSessions_version1.....	80
dcgmDeviceFbcSessions_version.....	80
dcgmDeviceVgpuEncSessions_version1.....	80
dcgmDeviceVgpuEncSessions_version.....	81
dcgmDeviceVgpuProcessUtilInfo_version1.....	81
dcgmDeviceVgpuProcessUtilInfo_version.....	81
dcgmDeviceVgpuls_version1.....	81
dcgmDeviceVgpuls_version.....	81
dcgmDeviceVgpuTypeInfo_version1.....	81
dcgmDeviceVgpuTypeInfo_version.....	81
dcgmDeviceAttributes_version1.....	81
dcgmDeviceAttributes_version.....	82
DCGM_MAX_VGPU_TYPES_PER_PGPU.....	82
dcgmVgpuDeviceAttributes_version5.....	82
dcgmVgpuDeviceAttributes_version.....	82
DCGM_DEVICE_UUID_BUFFER_SIZE.....	82
dcgmVgpuInstanceAttributes_version1.....	82
dcgmVgpuInstanceAttributes_version.....	82
dcgmConfig_version1.....	82
dcgmConfig_version.....	83
dcgmVgpuConfig_version1.....	83
dcgmVgpuConfig_version.....	83
dcgmPolicy_version1.....	83
dcgmPolicy_version.....	83

dcgmPolicyCallbackResponse_version1.....	83
dcgmPolicyCallbackResponse_version.....	83
DCGM_MAX_BLOB_LENGTH.....	83
dcgmFieldValue_version1.....	83
dcgmFieldValue_version2.....	84
dcgmFieldValue_version.....	84
DCGM_FV_FLAG_LIVE_DATA.....	84
DCGM_HEALTH_WATCH_COUNT_V1.....	84
DCGM_HEALTH_WATCH_COUNT_V2.....	84
dcgmHealthResponse_version1.....	84
dcgmHealthResponse_version2.....	84
dcgmHealthResponse_version.....	84
dcgmPidInfo_version1.....	84
dcgmPidInfo_version.....	85
dcgmJobInfo_version2.....	85
dcgmJobInfo_version.....	85
dcgmRunningProcess_version1.....	85
dcgmRunningProcess_version.....	85
dcgmDiagResponse_version3.....	85
dcgmDiagResponse_version.....	85
dcgmDeviceTopology_version1.....	85
dcgmDeviceTopology_version.....	85
dcgmGroupTopology_version1.....	86
dcgmGroupTopology_version.....	86
dcgmlntrospectContext_version1.....	86
dcgmlntrospectContext_version.....	86
dcgmlntrospectFieldsExecTime_version1.....	86
dcgmlntrospectFieldsExecTime_version.....	86
dcgmlntrospectFullFieldsExecTime_version1.....	86
dcgmlntrospectFullFieldsExecTime_version.....	87
dcgmlntrospectMemory_version1.....	87
dcgmlntrospectMemory_version.....	87
dcgmlntrospectFullMemory_version1.....	87
dcgmlntrospectFullMemory_version.....	87
dcgmlntrospectCpuUtil_version1.....	87
dcgmlntrospectCpuUtil_version.....	87
dcgmRunDiag_version1.....	87
dcgmRunDiag_version2.....	88
dcgmRunDiag_version.....	88
DCGM_GEGE_FLAG_ONLY_SUPPORTED.....	88
dcgmNvLinkStatus_version1.....	88
DCGM_MODULE_STATUSES_CAPACITY.....	88
dcgmModuleGetStatuses_version1.....	88

2.15. Field Types.....	88
DCGM_FT_BINARY.....	88
DCGM_FT_DOUBLE.....	88
DCGM_FT_INT64.....	89
DCGM_FT_STRING.....	89
DCGM_FT_TIMESTAMP.....	89
2.16. Field Scope.....	89
DCGM_FS_GLOBAL.....	89
DCGM_FS_ENTITY.....	89
DCGM_FS_DEVICE.....	89
DCGM_CUDA_COMPUTE_CAPABILITY_MAJOR.....	89
DCGM_CLOCKS_THROTTLE_REASON_GPU_IDLE.....	89
DCGM_CLOCKS_THROTTLE_REASON_CLOCKS_SETTING.....	90
DCGM_CLOCKS_THROTTLE_REASON_SW_POWER_CAP.....	90
DCGM_CLOCKS_THROTTLE_REASON_HW_SLOWDOWN.....	90
DCGM_CLOCKS_THROTTLE_REASON_SYNC_BOOST.....	90
DCGM_CLOCKS_THROTTLE_REASON_SW_THERMAL.....	91
DCGM_CLOCKS_THROTTLE_REASON_HW_THERMAL.....	91
DCGM_CLOCKS_THROTTLE_REASON_HW_POWER_BRAKE.....	91
DCGM_CLOCKS_THROTTLE_REASON_DISPLAY_CLOCKS.....	91
2.17. Field Entity.....	91
dcgm_field_entity_group_t.....	92
dcgm_field_eid_t.....	92
2.18. Field Identifiers.....	92
DcgmFieldGetById.....	92
DcgmFieldGetByTag.....	93
DcgmFieldsInit.....	93
DcgmFieldsTerm.....	93
DcgmFieldsGetEntityGroupString.....	93
DCGM_FI_UNKNOWN.....	94
DCGM_FI_DRIVER_VERSION.....	94
DCGM_FI_DEV_COUNT.....	94
DCGM_FI_DEV_NAME.....	94
DCGM_FI_DEV_BRAND.....	94
DCGM_FI_DEV_NVML_INDEX.....	94
DCGM_FI_DEV_SERIAL.....	94
DCGM_FI_DEV_UUID.....	94
DCGM_FI_DEV_MINOR_NUMBER.....	94
DCGM_FI_DEV_OEM_INFOROM_VER.....	94
DCGM_FI_DEV_PCI_BUSID.....	94
DCGM_FI_DEV_PCI_COMBINED_ID.....	95
DCGM_FI_DEV_PCI_SUBSYS_ID.....	95
DCGM_FI_GPU_TOPOLOGY_PCI.....	95

DCGM_FI_GPU_TOPOLOGY_NVLINK.....	95
DCGM_FI_GPU_TOPOLOGY_AFFINITY.....	95
DCGM_FI_DEV_CUDA_COMPUTE_CAPABILITY.....	95
DCGM_FI_DEV_COMPUTE_MODE.....	95
DCGM_FI_DEV_CPU_AFFINITY_0.....	95
DCGM_FI_DEV_CPU_AFFINITY_1.....	95
DCGM_FI_DEV_CPU_AFFINITY_2.....	95
DCGM_FI_DEV_CPU_AFFINITY_3.....	95
DCGM_FI_DEV_ECC_INFOROM_VER.....	96
DCGM_FI_DEV_POWER_INFOROM_VER.....	96
DCGM_FI_DEV_INFOROM_IMAGE_VER.....	96
DCGM_FI_DEV_INFOROM_CONFIG_CHECK.....	96
DCGM_FI_DEV_INFOROM_CONFIG_VALID.....	96
DCGM_FI_DEV_VBIOS_VERSION.....	96
DCGM_FI_DEV_BAR1_TOTAL.....	96
DCGM_FI_SYNC_BOOST.....	96
DCGM_FI_DEV_BAR1_USED.....	96
DCGM_FI_DEV_BAR1_FREE.....	96
DCGM_FI_DEV_SM_CLOCK.....	96
DCGM_FI_DEV_MEM_CLOCK.....	97
DCGM_FI_DEV_VIDEO_CLOCK.....	97
DCGM_FI_DEV_APP_SM_CLOCK.....	97
DCGM_FI_DEV_APP_MEM_CLOCK.....	97
DCGM_FI_DEV_CLOCK_THROTTLE_REASONS.....	97
DCGM_FI_DEV_AUTOBOOST.....	97
DCGM_FI_DEV_SUPPORTED_CLOCKS.....	97
DCGM_FI_DEV_MEMORY_TEMP.....	97
DCGM_FI_DEV_GPU_TEMP.....	97
DCGM_FI_DEV_POWER_USAGE.....	97
DCGM_FI_DEV_TOTAL_ENERGY_CONSUMPTION.....	97
DCGM_FI_DEV_SLOWDOWN_TEMP.....	98
DCGM_FI_DEV_SHUTDOWN_TEMP.....	98
DCGM_FI_DEV_POWER_MGMT_LIMIT.....	98
DCGM_FI_DEV_POWER_MGMT_LIMIT_MIN.....	98
DCGM_FI_DEV_POWER_MGMT_LIMIT_MAX.....	98
DCGM_FI_DEV_POWER_MGMT_LIMIT_DEF.....	98
DCGM_FI_DEV_ENFORCED_POWER_LIMIT.....	98
DCGM_FI_DEV_PSTATE.....	98
DCGM_FI_DEV_FAN_SPEED.....	98
DCGM_FI_DEV_PCIE_TX_THROUGHPUT.....	98
DCGM_FI_DEV_PCIE_RX_THROUGHPUT.....	98
DCGM_FI_DEV_PCIE_REPLAY_COUNTER.....	99
DCGM_FI_DEV_GPU_UTIL.....	99

DCGM_FI_DEV_MEM_COPY_UTIL.....	99
DCGM_FI_DEV_ACCOUNTING_DATA.....	99
DCGM_FI_DEV_ENC_UTIL.....	99
DCGM_FI_DEV_DEC_UTIL.....	99
DCGM_FI_DEV_MEM_COPY_UTIL_SAMPLES.....	99
DCGM_FI_DEV_GRAPHICS_PIDS.....	99
DCGM_FI_DEV_COMPUTE_PIDS.....	99
DCGM_FI_DEV_XID_ERRORS.....	99
DCGM_FI_DEV_PCIE_MAX_LINK_GEN.....	100
DCGM_FI_DEV_PCIE_MAX_LINK_WIDTH.....	100
DCGM_FI_DEV_PCIE_LINK_GEN.....	100
DCGM_FI_DEV_PCIE_LINK_WIDTH.....	100
DCGM_FI_DEV_POWER_VIOLATION.....	100
DCGM_FI_DEV_THERMAL_VIOLATION.....	100
DCGM_FI_DEV_SYNC_BOOST_VIOLATION.....	100
DCGM_FI_DEV_BOARD_LIMIT_VIOLATION.....	100
DCGM_FI_DEV_LOW_UTIL_VIOLATION.....	100
DCGM_FI_DEV_RELIABILITY_VIOLATION.....	100
DCGM_FI_DEV_TOTAL_APP_CLOCKS_VIOLATION.....	100
DCGM_FI_DEV_TOTAL_BASE_CLOCKS_VIOLATION.....	101
DCGM_FI_DEV_FB_TOTAL.....	101
DCGM_FI_DEV_FB_FREE.....	101
DCGM_FI_DEV_FB_USED.....	101
DCGM_FI_DEV_ECC_CURRENT.....	101
DCGM_FI_DEV_ECC_PENDING.....	101
DCGM_FI_DEV_ECC_SBE_VOL_TOTAL.....	101
DCGM_FI_DEV_ECC_DBE_VOL_TOTAL.....	101
DCGM_FI_DEV_ECC_SBE_AGG_TOTAL.....	101
DCGM_FI_DEV_ECC_DBE_AGG_TOTAL.....	101
DCGM_FI_DEV_ECC_SBE_VOL_L1.....	101
DCGM_FI_DEV_ECC_DBE_VOL_L1.....	102
DCGM_FI_DEV_ECC_SBE_VOL_L2.....	102
DCGM_FI_DEV_ECC_DBE_VOL_L2.....	102
DCGM_FI_DEV_ECC_SBE_VOL_DEV.....	102
DCGM_FI_DEV_ECC_DBE_VOL_DEV.....	102
DCGM_FI_DEV_ECC_SBE_VOL_REG.....	102
DCGM_FI_DEV_ECC_DBE_VOL_REG.....	102
DCGM_FI_DEV_ECC_SBE_VOL_TEX.....	102
DCGM_FI_DEV_ECC_DBE_VOL_TEX.....	102
DCGM_FI_DEV_ECC_SBE_AGG_L1.....	102
DCGM_FI_DEV_ECC_DBE_AGG_L1.....	102
DCGM_FI_DEV_ECC_SBE_AGG_L2.....	103
DCGM_FI_DEV_ECC_DBE_AGG_L2.....	103

DCGM_FI_DEV_ECC_SBE_AGG_DEV.....	103
DCGM_FI_DEV_ECC_DBE_AGG_DEV.....	103
DCGM_FI_DEV_ECC_SBE_AGG_REG.....	103
DCGM_FI_DEV_ECC_DBE_AGG_REG.....	103
DCGM_FI_DEV_ECC_SBE_AGG_TEX.....	103
DCGM_FI_DEV_ECC_DBE_AGG_TEX.....	103
DCGM_FI_DEV_RETIRE_SBE.....	103
DCGM_FI_DEV_RETIRE_DBE.....	103
DCGM_FI_DEV_RETIRE_PENDING.....	104
DCGM_FI_DEV_VIRTUAL_MODE.....	104
DCGM_FI_DEV_SUPPORTED_TYPE_INFO.....	104
DCGM_FI_DEV_CREATABLE_VGPU_TYPE_IDS.....	104
DCGM_FI_DEV_VGPU_INSTANCE_IDS.....	104
DCGM_FI_DEV_VGPU_UTILIZATIONS.....	104
DCGM_FI_DEV_VGPU_PER_PROCESS_UTILIZATION.....	104
DCGM_FI_DEV_ENC_STATS.....	104
DCGM_FI_DEV_FBC_STATS.....	104
DCGM_FI_DEV_FBC_SESSIONS_INFO.....	104
DCGM_FI_DEV_VGPU_VM_ID.....	104
DCGM_FI_DEV_VGPU_VM_NAME.....	105
DCGM_FI_DEV_VGPU_TYPE.....	105
DCGM_FI_DEV_VGPU_UUID.....	105
DCGM_FI_DEV_VGPU_DRIVER_VERSION.....	105
DCGM_FI_DEV_VGPU_MEMORY_USAGE.....	105
DCGM_FI_DEV_VGPU_LICENSE_STATUS.....	105
DCGM_FI_DEV_VGPU_FRAME_RATE_LIMIT.....	105
DCGM_FI_DEV_VGPU_ENC_STATS.....	105
DCGM_FI_DEV_VGPU_ENC_SESSIONS_INFO.....	105
DCGM_FI_DEV_VGPU_FBC_STATS.....	105
DCGM_FI_DEV_VGPU_FBC_SESSIONS_INFO.....	105
DCGM_FI_FIRST_VGPU_FIELD_ID.....	106
DCGM_FI_LAST_VGPU_FIELD_ID.....	106
DCGM_FI_MAX_VGPU_FIELDS.....	106
DCGM_FI_INTERNAL_FIELDS_0_START.....	106
DCGM_FI_INTERNAL_FIELDS_0_END.....	106
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P00.....	106
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P00.....	106
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P00.....	106
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P00.....	107
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P01.....	107
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P01.....	107
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P01.....	107
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P01.....	107

DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P02.....	107
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P02.....	107
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P02.....	108
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P02.....	108
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P03.....	108
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P03.....	108
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P03.....	108
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P03.....	108
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P04.....	108
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P04.....	109
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P04.....	109
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P04.....	109
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P05.....	109
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P05.....	109
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P05.....	109
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P05.....	109
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P06.....	110
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P06.....	110
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P06.....	110
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P06.....	110
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P07.....	110
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P07.....	110
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P07.....	110
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P07.....	110
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P08.....	111
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P08.....	111
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P08.....	111
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P08.....	111
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P09.....	111
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P09.....	111
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P09.....	111
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P09.....	112
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P10.....	112
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P10.....	112
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P10.....	112
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P10.....	112
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P11.....	112
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P11.....	112
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P11.....	113
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P11.....	113
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P12.....	113
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P12.....	113
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P12.....	113

DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P12.....	113
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P13.....	113
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P13.....	114
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P13.....	114
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P13.....	114
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P14.....	114
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P14.....	114
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P14.....	114
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P14.....	114
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P15.....	115
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P15.....	115
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P15.....	115
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P15.....	115
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P16.....	115
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P16.....	115
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P16.....	115
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P16.....	115
DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P17.....	116
DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P17.....	116
DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P17.....	116
DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P17.....	116
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P00.....	116
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P00.....	116
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P01.....	116
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P01.....	117
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P02.....	117
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P02.....	117
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P03.....	117
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P03.....	117
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P04.....	117
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P04.....	117
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P05.....	117
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P05.....	118
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P06.....	118
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P06.....	118
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P07.....	118
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P07.....	118
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P08.....	118
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P08.....	118
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P09.....	118
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P09.....	119
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P10.....	119
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P10.....	119

DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P11.....	119
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P11.....	119
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P12.....	119
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P12.....	119
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P13.....	119
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P13.....	120
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P14.....	120
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P14.....	120
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P15.....	120
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P15.....	120
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P16.....	120
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P16.....	120
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P17.....	120
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P17.....	121
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P00.....	121
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P00.....	121
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P01.....	121
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P01.....	121
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P02.....	121
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P02.....	121
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P03.....	122
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P03.....	122
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P04.....	122
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P04.....	122
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P05.....	122
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P05.....	122
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P06.....	122
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P06.....	122
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P07.....	123
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P07.....	123
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P08.....	123
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P08.....	123
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P09.....	123
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P09.....	123
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P10.....	123
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P10.....	123
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P11.....	124
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P11.....	124
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P12.....	124
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P12.....	124
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P13.....	124
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P13.....	124
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P14.....	124

DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P14.....	124
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P15.....	125
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P15.....	125
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P16.....	125
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P16.....	125
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P17.....	125
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P17.....	125
DCGM_FI_DEV_NVSWITCH_FATAL_ERRORS.....	125
DCGM_FI_DEV_NVSWITCH_NON_FATAL_ERRORS.....	125
DCGM_FI_FIRST_NVSWITCH_FIELD_ID.....	126
DCGM_FI_LAST_NVSWITCH_FIELD_ID.....	126
DCGM_FI_MAX_NVSWITCH_FIELDS.....	126
DCGM_FI_MAX_FIELDS.....	126
2.19. DCGMAPI_Admin_ExecCtrl.....	126
dcgmUpdateAllFields.....	126
dcgmPolicyTrigger.....	127
Chapter 3. Data Structures.....	128
dcgm_field_meta_t.....	130
dcgm_field_output_format_t.....	130
dcgmClockSet_v1.....	130
version.....	130
memClock.....	130
smClock.....	130
dcgmConfig_v1.....	130
version.....	131
gpuld.....	131
eccMode.....	131
computeMode.....	131
perfState.....	131
powerLimit.....	131
dcgmConfigPerfStateSettings_t.....	131
syncBoost.....	132
targetClocks.....	132
dcgmConfigPowerLimit_t.....	132
type.....	132
val.....	132
dcgmConnectV2Params_v1.....	132
version.....	132
persistAfterDisconnect.....	132
dcgmConnectV2Params_v2.....	133
version.....	133
persistAfterDisconnect.....	133
timeoutMs.....	133

addressIsUnixSocket.....	133
dcmDeviceAttributes_v1.....	133
version.....	134
clockSets.....	134
thermalSettings.....	134
powerLimits.....	134
identifiers.....	134
memoryUsage.....	134
unusedVgpuids.....	134
unusedActiveVgpuInstanceCount.....	134
unusedVgpuInstancelds.....	134
dcmDeviceEncStats_v1.....	134
version.....	135
sessionCount.....	135
averageFps.....	135
averageLatency.....	135
dcmDeviceFbcSessionInfo_v1.....	135
version.....	136
sessionId.....	136
pid.....	136
vgpuld.....	136
displayOrdinal.....	136
sessionType.....	136
sessionFlags.....	136
hMaxResolution.....	136
vMaxResolution.....	136
hResolution.....	136
vResolution.....	136
averageFps.....	137
averageLatency.....	137
dcmDeviceFbcSessions_v1.....	137
version.....	137
sessionCount.....	137
sessionInfo.....	137
dcmDeviceFbcStats_v1.....	137
version.....	138
sessionCount.....	138
averageFps.....	138
averageLatency.....	138
dcmDeviceIdentifiers_v1.....	138
version.....	139
brandName.....	139
deviceName.....	139

pciBusId.....	139
serial.....	139
uuid.....	139
vbios.....	139
inforomImageVersion.....	139
pciDeviceId.....	139
pciSubSystemId.....	139
driverVersion.....	139
virtualizationMode.....	139
dcgmDeviceMemoryUsage_v1.....	140
version.....	140
bar1Total.....	140
fbTotal.....	140
fbUsed.....	140
fbFree.....	140
dcgmDevicePidAccountingStats_v1.....	140
version.....	140
pid.....	140
gpuUtilization.....	140
memoryUtilization.....	141
maxMemoryUsage.....	141
startTimestamp.....	141
activeTimeUsec.....	141
dcgmDevicePowerLimits_v1.....	141
version.....	142
curPowerLimit.....	142
defaultPowerLimit.....	142
enforcedPowerLimit.....	142
minPowerLimit.....	142
maxPowerLimit.....	142
dcgmDeviceSupportedClockSets_v1.....	142
version.....	143
count.....	143
clockSet.....	143
dcgmDeviceThermals_v1.....	143
version.....	143
slowdownTemp.....	143
shutdownTemp.....	143
dcgmDeviceTopology_v1.....	143
version.....	143
cpuAffinityMask.....	143
numGpus.....	144
gpuld.....	144

path.....	144
localNvLinkIds.....	144
dcmDeviceVgpuEncSessions_v1.....	144
version.....	145
vgpuld.....	145
sessionId.....	145
pid.....	145
codecType.....	145
hResolution.....	145
vResolution.....	145
averageFps.....	145
averageLatency.....	145
dcmDeviceVgpuids_v1.....	145
version.....	146
unusedSupportedVgpuTypeCount.....	146
unusedSupportedVgpuTypeIds.....	146
unusedCreatableVgpuTypeCount.....	146
unusedCreatableVgpuTypeIds.....	146
dcmDeviceVgpuProcessUtilInfo_v1.....	146
version.....	147
vgpuld.....	147
vgpuProcessSamplesCount.....	147
pid.....	147
processName.....	147
smUtil.....	147
memUtil.....	147
encUtil.....	147
decUtil.....	147
dcmDeviceVgpuTypeInfo_v1.....	147
version.....	148
vgpuTypeInfo.....	148
vgpuTypeName.....	148
vgpuTypeClass.....	148
vgpuTypeLicense.....	148
deviceID.....	148
subsystemID.....	148
numDisplayHeads.....	148
maxInstances.....	148
frameRateLimit.....	148
maxResolutionX.....	148
maxResolutionY.....	148
fbTotal.....	148
dcmDeviceVgpuUtilInfo_v1.....	149

version.....	149
vgpuld.....	149
smUtil.....	149
memUtil.....	149
encUtil.....	149
decUtil.....	149
dcmDiagResponse_v3.....	149
version.....	150
gpuCount.....	150
blacklist.....	150
nvmlLibrary.....	150
cudaMainLibrary.....	150
cudaRuntimeLibrary.....	150
permissions.....	150
persistenceMode.....	150
environment.....	150
pageRetirement.....	150
inforom.....	150
graphicsProcesses.....	151
perGpuResponses.....	151
systemError.....	151
dcgmDiagResponsePerGpu_t.....	151
gpuld.....	151
hwDiagnosticReturn.....	151
results.....	151
dcgmErrorInfo_t.....	151
gpuld.....	152
fieldId.....	152
status.....	152
dcgmFieldGroupInfo_v1.....	152
version.....	152
numFieldIds.....	152
fieldGroupId.....	152
fieldGroupName.....	152
fieldIds.....	152
dcgmFieldValue_v1.....	152
version.....	153
fieldId.....	153
fieldType.....	153
status.....	153
ts.....	153
i64.....	153
dbl.....	153

str.....	153
blob.....	153
value.....	153
dcmFieldValue_v2.....	153
version.....	154
entityGroupId.....	154
entityId.....	154
fieldId.....	154
fieldType.....	154
status.....	154
unused.....	154
ts.....	154
i64.....	154
dbl.....	154
str.....	154
blob.....	154
value.....	154
dcmGpuUsageInfo_t.....	155
gpid.....	156
energyConsumed.....	156
powerUsage.....	156
pcieRxBandwidth.....	156
pcieTxBandwidth.....	156
pcieReplays.....	156
startTime.....	156
endTime.....	156
smUtilization.....	156
memoryUtilization.....	156
eccSingleBit.....	156
eccDoubleBit.....	157
memoryClock.....	157
smClock.....	157
numXidCriticalErrors.....	157
xidCriticalErrorsTs.....	157
numComputePids.....	157
computePidInfo.....	157
numGraphicsPids.....	157
graphicsPidInfo.....	157
maxGpuMemoryUsed.....	157
powerViolationTime.....	157
thermalViolationTime.....	158
reliabilityViolationTime.....	158
boardLimitViolationTime.....	158

lowUtilizationTime.....	158
syncBoostTime.....	158
overallHealth.....	158
system.....	158
health.....	158
dcmGroupEntityPair_t.....	158
entityGroupId.....	159
entityId.....	159
dcmGroupInfo_v1.....	159
version.....	159
count.....	159
gpuldList.....	159
groupName.....	159
dcmGroupInfo_v2.....	159
version.....	160
count.....	160
groupName.....	160
entityList.....	160
dcmGroupTopology_v1.....	160
version.....	160
groupCpuAffinityMask.....	160
numaOptimalFlag.....	160
slowestPath.....	161
dcmHealthResponse_v1.....	161
version.....	161
overallHealth.....	161
gpuCount.....	162
gpuld.....	162
incidentCount.....	162
system.....	162
health.....	162
errorString.....	162
dcmHealthResponse_v2.....	162
version.....	162
overallHealth.....	162
entityCount.....	163
entityGroupId.....	163
entityId.....	163
incidentCount.....	163
system.....	163
health.....	163
errorString.....	163
dcmIntrospectContext_v1.....	163

version.....	164
introspectLvl.....	164
fieldGroupId.....	164
fieldId.....	164
contextId.....	164
dcgmlIntrospectCpuUtil_v1.....	164
version.....	164
total.....	164
kernel.....	164
user.....	164
dcgmlIntrospectFieldsExecTime_v1.....	164
version.....	165
meanUpdateFreqUsec.....	165
recentUpdateUsec.....	165
totalEverUpdateUsec.....	165
dcgmlIntrospectFullFieldsExecTime_v1.....	165
version.....	166
aggregateInfo.....	166
hasGlobalInfo.....	166
globalInfo.....	166
gpuInfoCount.....	166
gpuidsForGpuInfo.....	166
gpuInfo.....	166
dcgmlIntrospectFullMemory_v1.....	166
version.....	167
aggregateInfo.....	167
hasGlobalInfo.....	167
globalInfo.....	167
gpuInfoCount.....	167
gpuidsForGpuInfo.....	167
gpuInfo.....	167
dcgmlIntrospectMemory_v1.....	167
version.....	168
bytesUsed.....	168
dcgmJobInfo_v2.....	168
version.....	168
numGpus.....	168
summary.....	168
gpus.....	168
dcgmModuleGetStatusesModule_t.....	168
id.....	169
status.....	169
dcgmNvLinkGpuLinkStatus_t.....	169

entityId.....	169
linkState.....	169
dcmgNvLinkNvSwitchLinkStatus_t.....	169
entityId.....	169
linkState.....	169
dcmgNvLinkStatus_v1.....	169
version.....	170
numGpus.....	170
gpus.....	170
numNvSwitches.....	170
nvSwitches.....	170
dcmgPidInfo_v1.....	170
version.....	171
pid.....	171
numGpus.....	171
summary.....	171
gpus.....	171
dcmgPidSingleInfo_t.....	171
gpuld.....	172
energyConsumed.....	172
pcieRxBandwidth.....	172
pcieTxBandwidth.....	172
pcieReplays.....	172
startTime.....	172
endTime.....	172
processUtilization.....	172
smUtilization.....	172
memoryUtilization.....	172
eccSingleBit.....	173
eccDoubleBit.....	173
memoryClock.....	173
smClock.....	173
numXidCriticalErrors.....	173
xidCriticalErrorsTs.....	173
numOtherComputePids.....	173
otherComputePids.....	173
numOtherGraphicsPids.....	173
otherGraphicsPids.....	173
maxGpuMemoryUsed.....	173
powerViolationTime.....	173
thermalViolationTime.....	174
reliabilityViolationTime.....	174
boardLimitViolationTime.....	174

lowUtilizationTime.....	174
syncBoostTime.....	174
overallHealth.....	174
system.....	174
health.....	174
dcmPolicy_v1.....	174
version.....	175
condition.....	175
mode.....	175
isolation.....	175
action.....	175
validation.....	175
response.....	175
parms.....	175
dcmPolicyCallbackResponse_v1.....	175
version.....	176
condition.....	176
dbe.....	176
pci.....	176
mpr.....	176
thermal.....	176
power.....	176
nvlink.....	176
xid.....	176
dcmPolicyConditionDbe_t.....	176
timestamp.....	177
location.....	177
numerrors.....	177
dcmPolicyConditionMpr_t.....	177
timestamp.....	177
sbepages.....	177
dbepages.....	177
dcmPolicyConditionNvlink_t.....	177
timestamp.....	178
fieldId.....	178
counter.....	178
dcmPolicyConditionParms_t.....	178
dcmPolicyConditionPci_t.....	178
timestamp.....	178
counter.....	178
dcmPolicyConditionPower_t.....	178
timestamp.....	179
powerViolation.....	179

dcgmPolicyConditionThermal_t.....	179
timestamp.....	179
thermalViolation.....	179
dcgmPolicyConditionXID_t.....	179
timestamp.....	179
errnum.....	179
dcgmPolicyViolationNotify_t.....	179
gpuld.....	180
violationOccurred.....	180
dcgmProcessUtilInfo_t.....	180
dcgmProcessUtilSample_t.....	180
dcgmRunningProcess_v1.....	180
version.....	180
pid.....	180
memoryUsed.....	180
dcgmStatSummaryFp64_t.....	180
minValue.....	181
maxValue.....	181
average.....	181
dcgmStatSummaryInt32_t.....	181
minValue.....	181
maxValue.....	181
average.....	181
dcgmStatSummaryInt64_t.....	181
minValue.....	182
maxValue.....	182
average.....	182
dcgmVgpuConfig_v1.....	182
version.....	183
gpuld.....	183
eccMode.....	183
computeMode.....	183
perfState.....	183
powerLimit.....	183
dcgmVgpuDeviceAttributes_v5.....	183
version.....	184
activeVgpuInstanceCount.....	184
activeVgpuInstancelds.....	184
creatableVgpuTypeCount.....	184
creatableVgpuTypelds.....	184
supportedVgpuTypeCount.....	184
supportedVgpuTypeInfo.....	184
vgpuUtilInfo.....	184

gpuUtil.....	184
memCopyUtil.....	184
encUtil.....	185
decUtil.....	185
dcmgVgpuInstanceAttributes_v1.....	185
version.....	186
vmlId.....	186
vmName.....	186
vgpuTypeId.....	186
vgpuUuid.....	186
vgpuDriverVersion.....	186
fbUsage.....	186
licenseStatus.....	186
frameRateLimit.....	186
Chapter 4. Data Fields.....	187

Chapter 1. CHANGE LOG

This chapter list changes in API that were introduced to the library.

1.3.0

- ▶ Field Groups, GPU Groups, and field watches created with a handle returned from `dcmConnect()` are now cleaned up upon disconnect. `dcmConnect_v2()` can be used to get the old behavior of objects persisting after disconnect.
- ▶ `dcmConnect_v2()` was added as a method for specifying additional connection options when connecting to the host engine.
- ▶ `dcmUnwatchFields()` was added as a method of unwatching fields that were previously watched with `dcmWatchFields()`
- ▶ `dcmActionValidate_v2()` was added to be able to pass more parameters to the DCGM GPU Diagnostic.
- ▶ `dcmDiagResponse_t` was increased from v2 to v3. See `dcmDiagResponse_v3` for details

1.2.3

- ▶ No API changes in this version.

1.1.1

- ▶ `dcm GetAllSupportedDevices()` was added as a method to get DCGM-supported GPU Ids. `dcm GetAllDevices()` can still be used to get all GPU Ids in the system.

1.0.0

- ▶ Initial Release.

Chapter 2. MODULES

Here is a list of all modules:

- ▶ Administrative
 - ▶ Init and Shutdown
- ▶ System
 - ▶ Discovery
 - ▶ Grouping
 - ▶ Field Grouping
 - ▶ Status handling
- ▶ Configuration
 - ▶ Setup and management
 - ▶ Manual Invocation
- ▶ Field APIs
- ▶ Process Statistics
- ▶ Job Statistics
- ▶ Health Monitor
- ▶ Policies
 - ▶ Setup and Management
 - ▶ Manual Invocation
- ▶ Topology
- ▶ Metadata
- ▶ Topology
- ▶ Modules
- ▶ Enums and Macros
- ▶ Structure definitions
- ▶ Field Types
- ▶ Field Scope

- ▶ [Field Entity](#)
- ▶ [Field Identifiers](#)
- ▶ [DCGM API Admin ExecCtrl](#)

2.1. Administrative

This chapter describes the administration interfaces for DCGM. It is the user's responsibility to call `dcmgInit()` before calling any other methods, and `dcmgShutdown()` once DCGM is no longer being used. The APIs in Administrative module can be broken down into following categories:

Init and Shutdown

2.1.1. Init and Shutdown

Administrative

Describes APIs to Initialize and Shutdown the DCGM Engine.

`dcmgReturn_t dcmgInit (void)`

Returns

- ▶ `DCGM_ST_OK` if DCGM has been properly initialized
- ▶ `DCGM_ST_INIT_ERROR` if there was an error initializing the library

Description

This method is used to initialize DCGM within this process. This must be called before `dcmgStartEmbedded()` or `dcmgConnect()`

*

`dcmgReturn_t dcmgShutdown (void)`

Returns

- ▶ `DCGM_ST_OK` if DCGM has been properly shut down
- ▶ `DCGM_ST_UNINITIALIZED` if the library was not shut down properly

Description

This method is used to shut down DCGM. Any embedded host engines or remote connections will automatically be shut down as well.

`dcgmReturn_t dcgmStartEmbedded (dcgmOperationMode_t opMode, dcgmHandle_t *pDcgmHandle)`

Parameters

opMode

IN : Collect data automatically or manually when asked by the user.

pDcgmHandle

OUT : DCGM Handle to use for API calls

Returns

- ▶ DCGM_ST_OK if DCGM was started successfully within our process
- ▶ DCGM_ST_UNINITIALIZED if DCGM has not been initialized with `dcmInit` yet

Description

Start an embedded host engine agent within this process.

The agent is loaded as a shared library. This mode is provided to avoid any extra jitter associated with an additional autonomous agent needs to be managed. In this mode, the user has to periodically call APIs such as `dcmPolicyTrigger` and `dcmUpdateAllFields` which tells DCGM to wake up and perform data collection and operations needed for policy management.

`dcgmReturn_t dcgmStopEmbedded (dcgmHandle_t pDcgmHandle)`

Parameters

pDcgmHandle

IN : DCGM Handle of the embedded host engine that came from `dcmStartEmbedded`

Returns

- ▶ DCGM_ST_OK if DCGM was stopped successfully within our process
- ▶ DCGM_ST_UNINITIALIZED if DCGM has not been initialized with `dcmInit` or the embedded host engine was not running.
- ▶ DCGM_ST_BADPARAM if an invalid parameter was provided
- ▶ DCGM_ST_INIT_ERROR if an error occurred while trying to start the host engine.

Description

Stop the embedded host engine within this process that was started with `dcmStartEmbedded`

dcgmReturn_t dcgmConnect (char *ipAddress, dcgmHandle_t *pDcgmHandle)

Parameters

ipAddress

IN : Valid IP address for the remote host engine to connect to. If ipAddress is specified as x.x.x.x it will attempt to connect to the default port specified by DCGM_HE_PORT_NUMBER If ipAddress is specified as x.x.x.x:yyyy it will attempt to connect to the port specified by yyyy

pDcgmHandle

OUT : DCGM Handle of the remote host engine

Returns

- ▶ DCGM_ST_OK if we successfully connected to the remote host engine
- ▶ DCGM_ST_CONNECTION_NOT_VALID if the remote host engine could not be reached
- ▶ DCGM_ST_UNINITIALIZED if DCGM has not been initialized with dcgmInit.
- ▶ DCGM_ST_BADPARAM if pDcgmHandle is NULL or ipAddress is invalid
- ▶ DCGM_ST_INIT_ERROR if DCGM encountered an error while initializing the remote client library
- ▶ DCGM_ST_UNINITIALIZED if DCGM has not been initialized with dcgmInit

Description

This method is used to connect to a stand-alone host engine process. Remote host engines are started by running the nv-hostengine command.

NOTE: dcgmConnect_v2 provides additional connection options.

dcgmReturn_t dcgmConnect_v2 (char *ipAddress, dcgmConnectV2Params_t *connectParams, dcgmHandle_t *pDcgmHandle)

Parameters

ipAddress

IN : Valid IP address for the remote host engine to connect to. If ipAddress is specified as x.x.x.x it will attempt to connect to the default port specified by DCGM_HE_PORT_NUMBER If ipAddress is specified as x.x.x.x:yyyy it will attempt to connect to the port specified by yyyy

connectParams

IN : Additional connection parameters. See [dcgmConnectV2Params_t](#) for details.

pDcgmHandle

OUT : DCGM Handle of the remote host engine

Returns

- ▶ DCGM_ST_OK if we successfully connected to the remote host engine
- ▶ DCGM_ST_CONNECTION_NOT_VALID if the remote host engine could not be reached
- ▶ DCGM_ST_UNINITIALIZED if DCGM has not been initialized with dcgmInit.
- ▶ DCGM_ST_BADPARAM if pDcgmHandle is NULL or ipAddress is invalid
- ▶ DCGM_ST_INIT_ERROR if DCGM encountered an error while initializing the remote client library
- ▶ DCGM_ST_UNINITIALIZED if DCGM has not been initialized with dcgmInit

Description

This method is used to connect to a stand-alone host engine process. Remote host engines are started by running the nv-hostengine command.

dcgmReturn_t dcgmDisconnect (dcgmHandle_t pDcgmHandle)**Parameters****pDcgmHandle**

IN: DCGM Handle that came from dcgmConnect

Returns

- ▶ DCGM_ST_OK if we successfully disconnected from the host engine
- ▶ DCGM_ST_UNINITIALIZED if DCGM has not been initialized with dcgmInit
- ▶ DCGM_ST_BADPARAM if pDcgmHandle is not a valid DCGM handle
- ▶ DCGM_ST_GENERIC_ERROR if an unspecified internal error occurred

Description

This method is used to disconnect from a stand-alone host engine process.

2.2. System

This chapter describes the APIs used to identify set of GPUs on the node, grouping functions to provide mechanism to operate on a group of GPUs, and status management APIs in order to get individual statuses for each operation. The APIs in System module can be broken down into following categories:

Discovery

Grouping

Field Grouping

Status handling

2.2.1. Discovery

System

The following APIs are used to discover GPUs and their attributes on a Node.

`dcgmReturn_t dcgmGetAllDevices (dcgmHandle_t pDcgmHandle,
unsigned int gpuidList, int *count)`

Parameters

`pDcgmHandle`

IN : DCGM Handle

`gpuidList`

OUT : Array reference to fill GPU Ids present on the system.

`count`

OUT : Number of GPUs returned in gpuidList.

Returns

- ▶ DCGM_ST_OK if the call was successful.
- ▶ DCGM_ST_BADPARAM if gpuidList or count were not valid.

Description

This method is used to get identifiers corresponding to all the devices on the system. The identifier represents DCGM GPU Id corresponding to each GPU on the system and is immutable during the lifespan of the engine. The list should be queried again if the engine is restarted.

The GPUs returned from this function include gpuids of GPUs that are not supported by DCGM. To only get gpuids of GPUs that are supported by DCGM, use `dcgmGetAllSupportedDevices()`.

dcgmReturn_t dcgmGetAllSupportedDevices (dcgmHandle_t pDcgmHandle, unsigned int gpuldList, int *count)

Parameters

pDcgmHandle

IN : DCGM Handle

gpuldList

OUT : Array reference to fill GPU Ids present on the system.

count

OUT : Number of GPUs returned in gpuldList.

Returns

- ▶ DCGM_ST_OK if the call was successful.
- ▶ DCGM_ST_BADPARAM if gpuldList or count were not valid.

Description

This method is used to get identifiers corresponding to all the DCGM-supported devices on the system. The identifier represents DCGM GPU Id corresponding to each GPU on the system and is immutable during the lifespan of the engine. The list should be queried again if the engine is restarted.

The GPUs returned from this function ONLY includes gpuids of GPUs that are supported by DCGM. To get gpuids of all GPUs in the system, use [dcgmGetAllDevices\(\)](#).

dcgmReturn_t dcgmGetDeviceAttributes (dcgmHandle_t pDcgmHandle, unsigned int gpuld, dcgmDeviceAttributes_t *pDcgmAttr)

Parameters

pDcgmHandle

IN : DCGM Handle

gpuld

IN : GPU Id corresponding to which the attributes should be fetched

pDcgmAttr

IN/OUT : Device attributes corresponding to gpuld. pDcgmAttr->version should be set to [dcgmDeviceAttributes_version](#) before this call.

Returns

- ▶ DCGM_ST_OK if the call was successful.

- ▶ DCGM_ST_VER_MISMATCH if pDcgmAttr->version is not set or is invalid.

Description

Gets device attributes corresponding to the gpuid. If operation is not successful for any of the requested fields then the field is populated with one of DCGM_BLANK_VALUES defined in dcgm_structs.h.

dcgmReturn_t dcgmGetEntityGroupEntities (dcgmHandle_t dcgmHandle, dcgm_field_entity_group_t entityGroup, dcgm_field_eid_t *entities, int *numEntities, unsigned int flags)

Parameters

dcgmHandle

IN: DCGM Handle

entityGroup

IN: Entity group to list entities of

entities

OUT: Array of entities for entityGroup

numEntities

IN/OUT: Upon calling, this should be the number of entities that entityList[] can hold.

Upon return, this will contain the number of entities actually saved to entityList.

flags

IN: Flags to modify the behavior of this request. See DCGM_GEGE_FLAG_* defines in dcgm_structs.h

Returns

- ▶ DCGM_ST_OK if the call was successful.
- ▶ DCGM_ST_INSUFFICIENT_SIZE if numEntities was not large enough to hold the number of entities in the entityGroup. numEntities will contain the capacity needed to complete this request successfully.
- ▶ DCGM_ST_NOT_SUPPORTED if the given entityGroup does not support enumeration.
- ▶ DCGM_ST_BADPARAM if any parameter is invalid

Description

Gets the list of entities that exist for a given entity group. This API can be used in place of [dcgmGetAllDevices](#).

dcgmReturn_t dcgmGetNvLinkLinkStatus (dcgmHandle_t dcgmHandle, dcgmNvLinkStatus_v1 *linkStatus)

Parameters

dcgmHandle

IN: DCGM Handle

linkStatus

OUT: Structure in which to store NvLink link statuses. .version should be set to dcgmNvLinkStatus_version1 before calling this.

Returns

- ▶ DCGM_ST_OK if the call was successful.
- ▶ DCGM_ST_NOT_SUPPORTED if the given entityGroup does not support enumeration.
- ▶ DCGM_ST_BADPARAM if any parameter is invalid

Description

Get the NvLink link status for every NvLink in this system. This includes the NvLinks of both GPUs and NvSwitches. Note that only NvSwitches and GPUs that are visible to the current environment will be returned in this structure.

2.2.2. Grouping

System

The following APIs are used for group management. The user can create a group of entities and perform an operation on a group of entities. If grouping is not needed and the user wishes to run commands on all GPUs seen by DCGM then the user can use DCGM_GROUP_ALL_GPUS or DCGM_GROUP_ALL_NVSWITCHES in place of group IDs when needed.

dcgmReturn_t dcgmGroupCreate (dcgmHandle_t pDcgmHandle, dcgmGroupType_t type, char *groupName, dcgmGpuGrp_t *pDcgmGrpId)

Parameters

pDcgmHandle

IN : DCGM Handle

type

IN : Type of Entity Group to be formed

groupName

IN : Desired name of the GPU group specified as NULL terminated C string

pDcgmGrpId

OUT : Reference to group ID

Returns

- ▶ DCGM_ST_OK if the group has been created
- ▶ DCGM_ST_BADPARAM if any of type, groupName, length or pDcgmGrpId is invalid
- ▶ DCGM_ST_MAX_LIMIT if number of groups on the system has reached the max limit DCGM_MAX_NUM_GROUPS
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized

Description

Used to create a entity group handle which can store one or more entity Ids as an opaque handle returned in pDcgmGrpId. Instead of executing an operation separately for each entity, the DCGM group enables the user to execute same operation on all the entities present in the group as a single API call.

To create the group with all the entities present on the system, the type field should be specified as DCGM_GROUP_DEFAULT or DCGM_GROUP_ALL_NVSWITCHES. To create an empty group, the type field should be specified as DCGM_GROUP_EMPTY. The empty group can be updated with the desired set of entities using the APIs [dcgmGroupAddDevice](#), [dcgmGroupAddEntity](#), [dcgmGroupRemoveDevice](#), and [dcgmGroupRemoveEntity](#).

dcgmReturn_t dcgmGroupDestroy (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId)

Parameters**pDcgmHandle**

IN : DCGM Handle

groupId

IN : Group ID

Returns

- ▶ DCGM_ST_OK if the group has been destroyed
- ▶ DCGM_ST_BADPARAM if groupId is invalid
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized
- ▶ DCGM_ST_NOT_CONFIGURED if entry corresponding to the group does not exists

Description

Used to destroy a group represented by groupId. Since DCGM group is a logical grouping of entities, the properties applied on the group stay intact for the individual entities even after the group is destroyed.

dcgmReturn_t dcgmGroupAddDevice (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, unsigned int gpuld)

Parameters

pDcgmHandle

IN : DCGM Handle

groupId

IN : Group Id to which device should be added

gpuld

IN : DCGM GPU Id

Returns

- ▶ DCGM_ST_OK if the GPU Id has been successfully added to the group
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized
- ▶ DCGM_ST_NOT_CONFIGURED if entry corresponding to the group (groupId) does not exists
- ▶ DCGM_ST_BADPARAM if gpuld is invalid or already part of the specified group

Description

Used to add specified GPU Id to the group represented by groupId.

dcgmReturn_t dcgmGroupAddEntity (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgm_field_entity_group_t entityGroupId, dcgm_field_eid_t entityId)

Parameters

pDcgmHandle

IN : DCGM Handle

groupId

IN : Group Id to which device should be added

entityGroupId

IN : Entity group that entityId belongs to

entityId

IN : DCGM entityId

Returns

- ▶ DCGM_ST_OK if the entity has been successfully added to the group
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized
- ▶ DCGM_ST_NOT_CONFIGURED if entry corresponding to the group (groupId) does not exists
- ▶ DCGM_ST_BADPARAM if entityId is invalid or already part of the specified group

Description

Used to add specified entity to the group represented by groupId.

dcgmReturn_t dcgmGroupRemoveDevice (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, unsigned int gpuld)

Parameters**pDcgmHandle**

IN : DCGM Handle

groupId

IN : Group ID from which device should be removed

gpuld

IN : DCGM GPU Id

Returns

- ▶ DCGM_ST_OK if the GPU Id has been successfully removed from the group
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized
- ▶ DCGM_ST_NOT_CONFIGURED if entry corresponding to the group (groupId) does not exists
- ▶ DCGM_ST_BADPARAM if gpuld is invalid or not part of the specified group

Description

Used to remove specified GPU Id from the group represented by groupId.

dcgmReturn_t dcgmGroupRemoveEntity (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgm_field_entity_group_t entityGroupId, dcgm_field_eid_t entityId)

Parameters**pDcgmHandle**

IN : DCGM Handle

groupId

IN : Group ID from which device should be removed

entityGroupId

IN : Entity group that entityId belongs to

entityId

IN : DCGM entityId

Returns

- ▶ DCGM_ST_OK if the entity has been successfully removed from the group
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized
- ▶ DCGM_ST_NOT_CONFIGURED if entry corresponding to the group (groupId) does not exists
- ▶ DCGM_ST_BADPARAM if entityId is invalid or not part of the specified group

Description

Used to remove specified entity from the group represented by groupId.

dcgmReturn_t dcgmGroupGetInfo (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmGroupInfo_t *pDcgmGroupInfo)

Parameters**pDcgmHandle**

IN : DCGM Handle

groupId

IN : Group ID for which information to be fetched

pDcgmGroupInfo

OUT : Group Information

Returns

- ▶ DCGM_ST_OK if the group info is successfully received.
- ▶ DCGM_ST_BADPARAM if any of groupId or pDcgmGroupInfo is invalid.
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized.
- ▶ DCGM_ST_MAX_LIMIT if the group does not contain the GPU
- ▶ DCGM_ST_NOT_CONFIGURED if entry corresponding to the group (groupId) does not exists

Description

Used to get information corresponding to the group represented by groupId. The information returned in pDcgmGroupInfo consists of group name, and the list of entities present in the group.

dcgmReturn_t dcgmGroupGetAllIds (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupIdList, unsigned int *count)

Parameters

pDcgmHandle

IN : DCGM Handle

groupIdList

OUT : List of Group Ids

count

OUT : The number of Group ids in the list

Returns

- ▶ DCGM_ST_OK if the ids of the groups were successfully retrieved
- ▶ DCGM_ST_BADPARAM if either of the groupIdList or count is null
- ▶ DCGM_ST_GENERIC_ERROR if an unknown error has occurred

Description

Used to get the Ids of all groups of entities. The information returned is a list of group ids in groupIdList as well as a count of how many ids there are in count. Please allocate enough memory for groupIdList. Memory of size MAX_NUM_GROUPS should be allocated for groupIdList.

2.2.3. Field Grouping

System

The following APIs are used for field group management. The user can create a group of fields and perform an operation on a group of fields at once.

dcgmReturn_t dcgmFieldGroupCreate (dcgmHandle_t dcgmHandle, int numFieldIds, unsigned short *fieldIds, char *fieldGroupName, dcgmFieldGrp_t *dcgmFieldGroupId)

Parameters

dcgmHandle

IN: DCGM handle

numFieldIds

IN: Number of field IDs that are being provided in fieldIds[]. Must be between 1 and DCGM_MAX_FIELD_IDS_PER_FIELD_GROUP.

fieldIds

IN: Field IDs to be added to the newly-created field group

fieldGroupName

IN: Unique name for this group of fields. This must not be the same as any existing field groups.

dcgmFieldGroupId

OUT: Handle to the newly-created field group

Returns

- ▶ DCGM_ST_OK if the field group was successfully created.
- ▶ DCGM_ST_BADPARAM if any parameters were bad
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized.
- ▶ DCGM_ST_MAX_LIMIT if too many field groups already exist

Description

Used to create a group of fields and return the handle in `dcgmFieldGroupId`

`dcgmReturn_t dcgmFieldGroupDestroy (dcgmHandle_t dcgmHandle, dcgmFieldGrp_t dcgmFieldGroupId)`

Parameters**dcgmHandle**

IN: DCGM handle

dcgmFieldGroupId

IN: Field group to remove

Returns

- ▶ DCGM_ST_OK if the field group was successfully removed
- ▶ DCGM_ST_BADPARAM if any parameters were bad
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized.

Description

Used to remove a field group that was created with `dcgmFieldGroupCreate`

dcgmReturn_t dcgmFieldGroupGetInfo (dcgmHandle_t dcgmHandle, dcgmFieldGroupInfo_t *fieldGroupInfo)

Parameters

dcgmHandle

IN: DCGM handle

fieldGroupInfo

IN/OUT: Info about all of the field groups that exist. .version should be set to [dcgmFieldGroupInfo_version](#) before this call .fieldGroupId should contain the fieldGroupId you are interested in querying information for.

Returns

- ▶ DCGM_ST_OK if the field group info was returned successfully
- ▶ DCGM_ST_BADPARAM if any parameters were bad
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized.
- ▶ DCGM_ST_VER_MISMATCH if .version is not set or is invalid.

Description

Used to get information about a field group that was created with [dcgmFieldGroupCreate](#).

dcgmReturn_t dcgmFieldGroupGetAll (dcgmHandle_t dcgmHandle, dcgmAllFieldGroup_t *allGroupInfo)

Parameters

dcgmHandle

IN: DCGM handle

allGroupInfo

IN/OUT: Info about all of the field groups that exist. .version should be set to [dcgmAllFieldGroup_version](#) before this call.

Returns

- ▶ DCGM_ST_OK if the field group info was successfully returned
- ▶ DCGM_ST_BADPARAM if any parameters were bad
- ▶ DCGM_ST_INIT_ERROR if the library has not been successfully initialized.
- ▶ DCGM_ST_VER_MISMATCH if .version is not set or is invalid.

Description

Used to get information about all field groups in the system.

2.2.4. Status handling

System

The following APIs are used to manage statuses for multiple operations on one or more GPUs.

`dcmReturn_t dcgmStatusCreate (dcgmStatus_t *statusHandle)`

Parameters

statusHandle

OUT : Reference to handle for list of statuses

Returns

- ▶ DCGM_ST_OK if the status handle is successfully created
- ▶ DCGM_ST_BADPARAM if statusHandle is invalid

Description

Creates reference to DCGM status handler which can be used to get the statuses for multiple operations on one or more devices.

The multiple statuses are useful when the operations are performed at group level. The status handle provides a mechanism to access error attributes for the failed operations.

The number of errors stored behind the opaque handle can be accessed using the API `dcgmStatusGetCount`. The errors are accessed from the opaque handle statusHandle using the API `dcgmStatusPopError`. The user can invoke `dcgmStatusPopError` for the number of errors or until all the errors are fetched.

When the status handle is not required any further then it should be deleted using the API `dcgmStatusDestroy`.

`dcmReturn_t dcgmStatusDestroy (dcgmStatus_t statusHandle)`

Parameters

statusHandle

IN : Handle to list of statuses

Returns

- ▶ DCGM_ST_OK if the status handle is successfully created
- ▶ DCGM_ST_BADPARAM if statusHandle is invalid

Description

Used to destroy status handle created using [dcgmStatusCreate](#).

dcgmReturn_t dcgmStatusGetCount (dcgmStatus_t statusHandle, unsigned int *count)

Parameters**statusHandle**

IN : Handle to list of statuses

count

OUT : Number of error entries present in the list of statuses

Returns

- ▶ DCGM_ST_OK if the error count is successfully received
- ▶ DCGM_ST_BADPARAM if any of statusHandle or count is invalid

Description

Used to get count of error entries stored inside the opaque handle statusHandle.

dcgmReturn_t dcgmStatusPopError (dcgmStatus_t statusHandle, dcgmErrorInfo_t *pDcgmErrorInfo)

Parameters**statusHandle**

IN : Handle to list of statuses

pDcgmErrorInfo

OUT : First error from the list of statuses

Returns

- ▶ DCGM_ST_OK if the error entry is successfully fetched
- ▶ DCGM_ST_BADPARAM if any of statusHandle or pDcgmErrorInfo is invalid
- ▶ DCGM_ST_NO_DATA if the status handle list is empty

Description

Used to iterate through the list of errors maintained behind statusHandle. The method pops the first error from the list of DCGM statuses. In order to iterate through all the errors, the user can invoke this API for the number of errors or until all the errors are fetched.

`dcmReturn_t dcgmStatusClear (dcgmStatus_t statusHandle)`

Parameters

statusHandle

IN : Handle to list of statuses

Returns

- ▶ DCGM_ST_OK if the errors are successfully cleared
- ▶ DCGM_ST_BADPARAM if statusHandle is invalid

Description

Used to clear all the errors in the status handle created by the API `dcgmStatusCreate`. After one set of operation, the statusHandle can be cleared and reused for the next set of operation.

2.3. Configuration

This chapter describes the methods that handle device configuration retrieval and default settings. The APIs in Configuration module can be broken down into following categories:

Setup and management

Manual Invocation

2.3.1. Setup and management

Configuration

Describes APIs to Get/Set configuration on the group of GPUs.

```
dcgmReturn_t dcgmConfigSet (dcgmHandle_t pDcgmHandle,
dcgmGpuGrp_t groupId, dcgmConfig_t *pDeviceConfig,
dcgmStatus_t statusHandle)
```

Parameters

pDcgmHandle

IN : DCGM Handle

groupId

IN : Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group.

pDeviceConfig

IN : Pointer to memory to hold desired configuration to be applied for all the GPU in the group represented by groupId. The caller must populate the version field of pDeviceConfig.

statusHandle

IN/OUT : Resulting error status for multiple operations. Pass it as NULL if the detailed error information is not needed. Look at [dcgmStatusCreate](#) for details on creating status handle.

Returns

- ▶ DCGM_ST_OK if the configuration has been successfully set.
- ▶ DCGM_ST_BADPARAM if any of groupId or pDeviceConfig is invalid.
- ▶ DCGM_ST_VER_MISMATCH if pDeviceConfig has the incorrect version.
- ▶ DCGM_ST_GENERIC_ERROR if an unknown error has occurred.

Description

Used to set configuration for the group of one or more GPUs identified by groupId.

The configuration settings specified in pDeviceConfig are applied to all the GPUs in the group. Since DCGM group is a logical grouping of GPUs, the configuration settings stays intact for the individual GPUs even after the group is destroyed.

If the user wishes to ignore the configuration of one or more properties in the input pDeviceConfig then the property should be specified as one of DCGM_INT32_BLANK, DCGM_INT64_BLANK, DCGM_FP64_BLANK or DCGM_STR_BLANK based on the data type of the property to be ignored.

If any of the properties fail to be configured for any of the GPUs in the group then the API returns an error. The status handle statusHandle should be further evaluated to access error attributes for the failed operations. Please refer to status management APIs at [Status handling](#) to access the error attributes.

To find out valid supported clock values that can be passed to `dcmConfigSet`, look at the device attributes of a GPU in the group using the API `dcmGetDeviceAttributes`.

`dcmReturn_t dcmConfigGet (dcmHandle_t pDcmHandle, dcmGpuGrp_t groupId, dcmConfigType_t type, int count, dcmConfig_t deviceConfigList, dcmStatus_t statusHandle)`

Parameters

pDcmHandle

IN : DCGM Handle

groupId

IN : Group ID representing collection of one or more GPUs. Look at `dcmGroupCreate` for details on creating the group.

type

IN : Type of configuration values to be fetched.

count

IN : The number of entries that deviceConfigList array can store.

deviceConfigList

OUT : Pointer to memory to hold requested configuration corresponding to all the GPUs in the group (groupId). The size of the memory must be greater than or equal to hold output information for the number of GPUs present in the group (groupId).

statusHandle

IN/OUT : Resulting error status for multiple operations. Pass it as NULL if the detailed error information is not needed. Look at `dcmStatusCreate` for details on creating status handle.

Returns

- ▶ DCGM_ST_OK if the configuration has been successfully fetched.
- ▶ DCGM_ST_BADPARAM if any of groupId, type, count, or deviceConfigList is invalid.
- ▶ DCGM_ST_NOT_CONFIGURED if the target configuration is not already set.
- ▶ DCGM_ST_VER_MISMATCH if deviceConfigList has the incorrect version.
- ▶ DCGM_ST_GENERIC_ERROR if an unknown error has occurred.

Description

Used to get configuration for all the GPUs present in the group.

This API can get the most recent target or desired configuration set by `dcmConfigSet`. Set type as DCGM_CONFIG_TARGET_STATE to get target configuration. The target configuration properties are maintained by DCGM and are automatically enforced after a GPU reset or reinitialization is completed.

The method can also be used to get the actual configuration state for the GPUs in the group. Set type as DCGM_CONFIG_CURRENT_STATE to get the actually configuration state. Ideally, the actual configuration state will be exact same as the target configuration state.

If any of the property in the target configuration is unknown then the property value in the output is populated as one of DCGM_INT32_BLANK, DCGM_INT64_BLANK, DCGM_FP64_BLANK or DCGM_STR_BLANK based on the data type of the property.

If any of the property in the current configuration state is not supported then the property value in the output is populated as one of DCGM_INT32_NOT_SUPPORTED, DCGM_INT64_NOT_SUPPORTED, DCGM_FP64_NOT_SUPPORTED or DCGM_STR_NOT_SUPPORTED based on the data type of the property.

If any of the properties can't be fetched for any of the GPUs in the group then the API returns an error. The status handle statusHandle should be further evaluated to access error attributes for the failed operations. Please refer to status management APIs at [Status handling](#) to access the error attributes.

2.3.2. Manual Invocation

Configuration

Describes APIs used to manually enforce the desired configuration on a group of GPUs.

dcgmReturn_t dcgmConfigEnforce (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmStatus_t statusHandle)

Parameters

pDcgmHandle

IN : DCGM Handle

groupId

IN : Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

statusHandle

IN/OUT : Resulting error status for multiple operations. Pass it as NULL if the detailed error information is not needed. Look at [dcgmStatusCreate](#) for details on creating status handle.

Returns

- ▶ DCGM_ST_OK if the configuration has been successfully enforced.
- ▶ DCGM_ST_BADPARAM if groupId is invalid.
- ▶ DCGM_ST_NOT_CONFIGURED if the target configuration is not already set.

- ▶ DCGM_ST_GENERIC_ERROR if an unknown error has occurred.

Description

Used to enforce previously set configuration for all the GPUs present in the group.

This API provides a mechanism to the users to manually enforce the configuration at any point of time. The configuration can only be enforced if it's already configured using the API [dcgmConfigSet](#).

If any of the properties can't be enforced for any of the GPUs in the group then the API returns an error. The status handle statusHandle should be further evaluated to access error attributes for the failed operations. Please refer to status management APIs at [Status handling](#) to access the error attributes.

2.4. Field APIs

These APIs are responsible for watching, unwatching, and updating specific fields as defined by DCGM_FI_*

```
dcgmReturn_t dcgmWatchFields (dcgmHandle_t pDcmHandle, dcgmGpuGrp_t groupId, dcgmFieldGrp_t fieldGroupId, long long updateFreq, double maxKeepAge, int maxKeepSamples)
```

Parameters

pDcmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more entities. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs or DCGM_GROUP_ALL_NVSWITCHES to perform the operation on all NvSwitches.

fieldGroupId

IN: Fields to watch.

updateFreq

IN: How often to update this field in usec

maxKeepAge

IN: How long to keep data for this field in seconds

maxKeepSamples

IN: Maximum number of samples to keep. 0=no limit

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid

Description

Request that DCGM start recording updates for a given field collection.

Note that the first update of the field will not occur until the next field update cycle. To force a field update cycle, call `dcmUpdateAllFields(1)`.

`dcmReturn_t dcgmUnwatchFields (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmFieldGrp_t fieldGroupId)`

Parameters**pDcgmHandle**

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more entities. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs or DCGM_GROUP_ALL_NVSWITCHES to to perform the operation on all NvSwitches.

fieldGroupId

IN: Fields to unwatch.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid

Description

Request that DCGM stop recording updates for a given field collection.

`dcmReturn_t dcgmGetValuesSince (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmFieldGrp_t fieldGroupId, long long sinceTimestamp, long long`

`*nextSinceTimestamp, dcgmFieldValueEnumeration_f
enumCB, void *userData)`

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

fieldGroupId

IN: Fields to return data for

sinceTimestamp

IN: Timestamp to request values since in usec since 1970. This will be returned in nextSinceTimestamp for subsequent calls 0 = request all data

nextSinceTimestamp

OUT: Timestamp to use for sinceTimestamp on next call to this function

enumCB

IN: Callback to invoke for every field value update. Note that multiple updates can be returned in each invocation

userData

IN: User data pointer to pass to the userData field of enumCB.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_NOT_SUPPORTED if one of the entities was from a non-GPU type
- ▶ DCGM_ST_BADPARAM if a parameter is invalid

Description

Request updates for all field values that have updated since a given timestamp

This version only works with GPU entities. Use [dcgmGetValuesSince_v2](#) for entity groups containing NvSwitches.

dcgmReturn_t dcgmGetValuesSince_v2
(dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t
groupId, dcgmFieldGrp_t fieldGroupId, long long
sinceTimestamp, long long *nextSinceTimestamp,

`dcmFieldValueEntityEnumeration_f enumCB, void *userData)`

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more entities. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs or DCGM_GROUP_ALL_NVSWITCHES to perform the operation on all NvSwitches.

fieldGroupId

IN: Fields to return data for

sinceTimestamp

IN: Timestamp to request values since in usec since 1970. This will be returned in nextSinceTimestamp for subsequent calls 0 = request all data

nextSinceTimestamp

OUT: Timestamp to use for sinceTimestamp on next call to this function

enumCB

IN: Callback to invoke for every field value update. Note that multiple updates can be returned in each invocation

userData

IN: User data pointer to pass to the userData field of enumCB.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid

Description

Request updates for all field values that have updated since a given timestamp

This version works with non-GPU entities like NvSwitches

`dcmReturn_t dcgmGetLatestValues (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmFieldGrp_t`

**fieldGroupId, dcgmFieldValueEnumeration_f enumCB,
void *userData)**

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

fieldGroupId

IN: Fields to return data for.

enumCB

IN: Callback to invoke for every field value update. Note that multiple updates can be returned in each invocation

userData

IN: User data pointer to pass to the userData field of enumCB.

Description

Request latest cached field value for a field value collection

This version only works with GPU entities. Use [dcgmGetLatestValues_v2](#) for entity groups containing NvSwitches.

- ▶ `DCGM_ST_OK` if the call was successful
- ▶ `DCGM_ST_NOT_SUPPORTED` if one of the entities was from a non-GPU type
- ▶ `DCGM_ST_BADPARAM` if a parameter is invalid

**dcgmReturn_t dcgmGetLatestValues_v2 (dcgmHandle_t
pDcgmHandle, dcgmGpuGrp_t groupId, dcgmFieldGrp_t
fieldGroupId, dcgmFieldValueEntityEnumeration_f
enumCB, void *userData)**

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more entities. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs or DCGM_GROUP_ALL_NVSWITCHES to perform the operation on all NvSwitches.

fieldGroupId

IN: Fields to return data for.

enumCB

IN: Callback to invoke for every field value update. Note that multiple updates can be returned in each invocation

userData

IN: User data pointer to pass to the userData field of enumCB.

Description

Request latest cached field value for a field value collection

This version works with non-GPU entities like NvSwitches

- ▶ **DCGM_ST_OK** if the call was successful
- ▶ **DCGM_ST_NOT_SUPPORTED** if one of the entities was from a non-GPU type
- ▶ **DCGM_ST_BADPARAM** if a parameter is invalid

dcmReturn_t dcgmGetLatestValuesForFields

(dcgmHandle_t pDcgmHandle, int gpuld, unsigned short fields, unsigned int count, dcgmFieldValue_v1 values)

Parameters**pDcgmHandle**

IN: DCGM Handle

gpuId

IN: Gpu ID representing the GPU for which the fields are being requested.

fields

IN: Field IDs to return data for. See the definitions in dcgm_fields.h that start with **DCGM_FI_**.

count

IN: Number of field IDs in fields[] array.

values

OUT: Latest field values for the fields in fields[].

Description

Request latest cached field value for a GPU

dcmReturn_t dcgmEntityGetLatestValues

(dcgmHandle_t pDcgmHandle,

dcgm_field_entity_group_t entityGroup, int

**entityId, unsigned short fields, unsigned int count,
dcgmFieldValue_v1 values)**

Parameters

pDcgmHandle

IN: DCGM Handle

entityGroup

IN: entity_group_t (e.g. switch)

entityId

IN: entity ID representing the rntity for which the fields are being requested.

fields

IN: Field IDs to return data for. See the definitions in dcgm_fields.h that start with DCGM_FI_.

count

IN: Number of field IDs in fields[] array.

values

OUT: Latest field values for the fields in fields[].

Description

Request latest cached field value for a group of fields for a specific entity

**dcgmReturn_t dcgmEntitiesGetLatestValues
(dcgmHandle_t pDcgmHandle, dcgmGroupEntityPair_t
entities, unsigned int entityCount, unsigned short
fields, unsigned int fieldCount, unsigned int flags,
dcgmFieldValue_v2 values)**

Parameters

pDcgmHandle

IN: DCGM Handle

entities

IN: List of entities to get values for

entityCount

IN: Number of entries in entities[]

fields

IN: Field IDs to return data for. See the definitions in dcgm_fields.h that start with DCGM_FI_.

fieldCount

IN: Number of field IDs in fields[] array.

flags

IN: Optional flags that affect how this request is processed. Pass `DCGM_FV_FLAG_LIVE_DATA` here to retrieve a live driver value rather than a cached value. See that flag's documentation for caveats.

values

OUT: Latest field values for the fields requested. This must be able to hold `entityCount * fieldCount` field value records.

Description

Request the latest cached or live field value for a list of fields for a group of entities

Note: The returned entities are not guaranteed to be in any order. Reordering can occur internally in order to optimize calls to the NVIDIA driver.

2.5. Process Statistics

Describes APIs to investigate statistics such as accounting, performance and errors during the lifetime of a GPU process

`dcmReturn_t dcgmWatchPidFields (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, long long updateFreq, double maxKeepAge, int maxKeepSamples)`

Parameters**pDcgmHandle**

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more GPUs. Look at `dcgmGroupCreate` for details on creating the group. Alternatively, pass in the group id as `DCGM_GROUP_ALL_GPUS` to perform operation on all the GPUs.

updateFreq

IN: How often to update this field in usec

maxKeepAge

IN: How long to keep data for this field in seconds

maxKeepSamples

IN: Maximum number of samples to keep. 0=no limit

Returns

- ▶ `DCGM_ST_OK` if the call was successful
- ▶ `DCGM_ST_BADPARAM` if a parameter is invalid

- ▶ DCGM_ST_REQUIRES_ROOT if the host engine is being run as non-root, and accounting mode could not be enabled (requires root). Run "nvidia-smi -am 1" as root on the node before starting DCGM to fix this.

Description

Request that DCGM start recording stats for fields that can be queried with [dcgmGetPidInfo\(\)](#).

Note that the first update of the field will not occur until the next field update cycle. To force a field update cycle, call `dcgmUpdateAllFields(1)`.

dcgmReturn_t dcgmGetPidInfo (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmPidInfo_t *pidInfo)

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

pidInfo

IN/OUT: Structure to return information about pid in. `pidInfo->pid` must be set to the pid in question. `pidInfo->version` should be set to `dcgmPidInfo_version`.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_NO_DATA if the PID did not run on any GPU

Description

Get information about all GPUs while the provided pid was running

In order for this request to work, you must first call [dcgmWatchPidFields\(\)](#) to make sure that DCGM is watching the appropriate field IDs that will be populated in `pidInfo`

2.6. Job Statistics

The client can invoke DCGM APIs to start and stop collecting the stats at the process boundaries (during prologue and epilogue). This will enable DCGM to monitor all the PIDs while the job is in progress, and provide a summary of active processes and resource usage during the window of interest.

dcgmReturn_t dcgmWatchJobFields (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, long long updateFreq, double maxKeepAge, int maxKeepSamples)

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

updateFreq

IN: How often to update this field in usec

maxKeepAge

IN: How long to keep data for this field in seconds

maxKeepSamples

IN: Maximum number of samples to keep. 0=no limit

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid
- ▶ DCGM_ST_REQUIRES_ROOT if the host engine is being run as non-root, and accounting mode could not be enabled (requires root). Run "nvidia-smi -am 1" as root on the node before starting DCGM to fix this.

Description

Request that DCGM start recording stats for fields that are queried with [dcgmJobGetStats\(\)](#)

Note that the first update of the field will not occur until the next field update cycle. To force a field update cycle, call [dcgmUpdateAllFields\(1\)](#).

dcgmReturn_t dcgmJobStartStats (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, char jobId)

Parameters

pDcgmHandle

IN : DCGM Handle

groupId

IN : Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

jobId

IN : User provided string to represent the job

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid
- ▶ DCGM_ST_DUPLICATE_KEY if the specified jobId is already in use

Description

This API is used by the client to notify DCGM about the job to be started. Should be invoked as part of job prologue

dcgmReturn_t dcgmJobStopStats (dcgmHandle_t pDcgmHandle, char jobId)

Parameters

pDcgmHandle

IN : DCGM Handle

jobId

IN : User provided string to represent the job

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid
- ▶ DCGM_ST_NO_DATA if jobId is not a valid job identifier.

Description

This API is used by the clients to notify DCGM to stop collecting stats for the job represented by job id. Should be invoked as part of job epilogue. The job Id remains available to view the stats at any point but cannot be used to start a new job. You must call `dcmWatchJobFields()` before this call to enable watching of job

dcmReturn_t dcmJobGetStats (dcgmHandle_t pDcmHandle, char jobId, dcgmJobInfo_t *pJobInfo)

Parameters

pDcmHandle

IN : DCGM Handle

jobId

IN : User provided string to represent the job

pJobInfo

IN/OUT : Structure to return information about the job. .version should be set to `dcmJobInfo_version` before this call.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid
- ▶ DCGM_ST_NO_DATA if jobId is not a valid job identifier.
- ▶ DCGM_ST_VER_MISMATCH if .version is not set or is invalid.

Description

Get stats for the job identified by DCGM generated job id. The stats can be retrieved at any point when the job is in process. If you want to reuse this jobId, call `dcmJobRemove` after this call.

dcmReturn_t dcmJobRemove (dcgmHandle_t pDcmHandle, char jobId)

Parameters

pDcmHandle

IN : DCGM Handle

jobId

IN : User provided string to represent the job

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid
- ▶ DCGM_ST_NO_DATA if jobId is not a valid job identifier.

Description

This API tells DCGM to stop tracking the job given by jobId. After this call, you will no longer be able to call [dcgmJobGetStats\(\)](#) on this jobId. However, you will be able to reuse jobId after this call.

dcgmReturn_t dcgmJobRemoveAll (dcgmHandle_t pDcgmHandle)

Parameters**pDcgmHandle**

IN : DCGM Handle

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid

Description

This API tells DCGM to stop tracking all jobs. After this call, you will no longer be able to call [dcgmJobGetStats\(\)](#) on any jobs until you call dcgmJobStartStats again. You will be able to reuse any previously-used jobIds after this call.

2.7. Health Monitor

This chapter describes the methods that handle the GPU health monitor.

`dcgmReturn_t dcgmHealthSet (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmHealthSystems_t systems)`

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more entities. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs or DCGM_GROUP_ALL_NVSWITCHES to perform operation on all the NvSwitches.

systems

IN: An enum representing systems that should be enabled for health checks logically OR'd together. Refer to [dcgmHealthSystems_t](#) for details.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid

Description

Enable the DCGM health check system for the given systems defined in [dcgmHealthSystems_t](#)

`dcgmReturn_t dcgmHealthGet (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmHealthSystems_t *systems)`

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more entities. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs or DCGM_GROUP_ALL_NVSWITCHES to perform operation on all the NvSwitches.

systems

OUT: An integer representing the enabled systems for the given group Refer to [dcgmHealthSystems_t](#) for details.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid

Description

Retrieve the current state of the DCGM health check system

dcgmReturn_t dcgmHealthCheck (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmHealthResponse_t *results)

Parameters**pDcgmHandle**

IN: DCGM Handle

groupId

IN: Group ID representing a collection of one or more entities. Refer to [dcgmGroupCreate](#) for details on creating a group

results

OUT: A reference to the dcgmHealthResponse_t structure to populate. results->version must be set to dcgmHealthResponse_version.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if a parameter is invalid
- ▶ DCGM_ST_VER_MISMATCH if results->version is not dcgmHealthResponse_version

Description

Check the configured watches for any errors/failures/warnings that have occurred since the last time this check was invoked. On the first call, stateful information about all of the enabled watches within a group is created but no error results are provided. On subsequent calls, any error information will be returned.

2.8. Policies

This chapter describes the methods that handle system policy management and violation settings. The APIs in Policies module can be broken down into following categories:

Setup and Management

Manual Invocation

2.8.1. Setup and Management

Policies

Describes APIs for setting up policies and registering callbacks to receive notification in case specific policy condition has been violated.

`dcgmReturn_t dcgmPolicySet (dcgmHandle_t pDcgmHandle,
dcgmGpuGrp_t groupId, dcgmPolicy_t *policy, dcgmStatus_t
statusHandle)`

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

policy

IN: A reference to [dcgmPolicy_t](#) that will be applied to all GPUs in the group.

statusHandle

IN/OUT: Resulting status for the operation. Pass it as NULL if the detailed error information is not needed. Refer to [dcgmStatusCreate](#) for details on creating a status handle.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if groupId or policy is invalid
- ▶ DCGM_ST_NOT_SUPPORTED if any non-Tesla GPUs are part of the GPU group specified in groupId

- ▶ DCGM_ST_* a different error has occurred and is stored in statusHandle. Refer to [dcgmReturn_t](#)

Description

Set the current violation policy inside the policy manager. Given the conditions within the [dcgmPolicy_t](#) structure, if a violation has occurred, subsequent action(s) may be performed to either report or contain the failure.

This API is only supported on Tesla GPUs and will return DCGM_ST_NOT_SUPPORTED if any non-Tesla GPUs are part of the GPU group specified in groupId.

```
dcgmReturn_t dcgmPolicyGet (dcgmHandle_t pDcgmHandle,
dcgmGpuGrp_t groupId, int count, dcgmPolicy_t *policy,
dcgmStatus_t statusHandle)
```

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

count

IN: The size of the policy array. This is the maximum number of policies that will be retrieved and ultimately should correspond to the number of GPUs specified in the group.

policy

OUT: A reference to [dcgmPolicy_t](#) that will used as storage for the current policies applied to each GPU in the group.

statusHandle

IN/OUT: Resulting status for the operation. Pass it as NULL if the detailed error information for the operation is not needed. Refer to [dcgmStatusCreate](#) for details on creating a status handle.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if groupId or policy is invalid
- ▶ DCGM_ST_* a different error has occurred and is stored in statusHandle. Refer to [dcgmReturn_t](#)

Description

Get the current violation policy inside the policy manager. Given a groupId, a number of policy structures are retrieved.

dcgmReturn_t dcgmPolicyRegister (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmPolicyCondition_t condition, fpRecvUpdates beginCallback, fpRecvUpdates finishCallback)

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

condition

IN: The set of conditions specified as an OR'd list (see [dcgmPolicyCondition_t](#)) for which to register a callback function

beginCallback

IN: A reference to a function that should be called should a violation occur. This function will be called prior to any actions specified by the policy are taken.

finishCallback

IN: A reference to a function that should be called should a violation occur. This function will be called after any action specified by the policy are completed.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if groupId, condition, is invalid, beginCallback, or finishCallback is NULL
- ▶ DCGM_ST_NOT_SUPPORTED if any non-Tesla GPUs are part of the GPU group specified in groupId

Description

Register a function to be called when a specific policy condition (see [dcgmPolicyCondition_t](#)) has been violated. This callback(s) will be called automatically when in DCGM_OPERATION_MODE_AUTO mode and only after [dcgmPolicyTrigger](#) when in DCGM_OPERATION_MODE_MANUAL mode. All callbacks are made within a separate thread.

This API is only supported on Tesla GPUs and will return DCGM_ST_NOT_SUPPORTED if any non-Tesla GPUs are part of the GPU group specified in groupId.

`dcmReturn_t dcgmPolicyUnregister (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmPolicyCondition_t condition)`

Parameters

`pDcgmHandle`

IN: DCGM Handle

`groupId`

IN: Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

`condition`

IN: The set of conditions specified as an OR'd list (see [dcgmPolicyCondition_t](#)) for which to unregister a callback function

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if groupId, condition, is invalid or callback is NULL

Description

Unregister a function to be called for a specific policy condition (see [dcgmPolicyCondition_t](#)). This function will unregister all callbacks for a given condition and handle.

2.8.2. Manual Invocation

Policies

Describes APIs which can be used to perform direct actions (e.g. Perform GPU Reset, Run Health Diagnostics) on a group of GPUs.

```
dcgmReturn_t dcgmActionValidate (dcgmHandle_t pDcgmHandle,
dcgmGpuGrp_t groupId, dcgmPolicyValidation_t validate,
dcgmDiagResponse_t *response)
```

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

validate

IN: The validation to perform after the action.

response

OUT: Result of the validation process. Refer to [dcgmDiagResponse_t](#) for details.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_NOT_SUPPORTED if running the specified validate is not supported. This is usually due to the Tesla recommended driver not being installed on the system.
- ▶ DCGM_ST_BADPARAM if groupId, validate, or statusHandle is invalid
- ▶ DCGM_ST_GENERIC_ERROR an internal error has occurred
- ▶ DCGM_ST_GROUP_INCOMPATIBLE if groupId refers to a group of non-homogeneous GPUs. This is currently not allowed.

Description

Inform the action manager to perform a manual validation of a group of GPUs on the system

***** DEPRECATED *****

```
dcgmReturn_t dcgmActionValidate_v2 (dcgmHandle_t pDcgmHandle,
dcgmRunDiag_t *drd, dcgmDiagResponse_t *response)
```

Parameters

pDcgmHandle

IN: DCGM Handle

drd

IN: Contains the group id, test names, test parameters, struct version, and the validation that should be performed. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

response

OUT: Result of the validation process. Refer to [dcgmDiagResponse_t](#) for details.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_NOT_SUPPORTED if running the specified validate is not supported. This is usually due to the Tesla recommended driver not being installed on the system.
- ▶ DCGM_ST_BADPARAM if groupId, validate, or statusHandle is invalid
- ▶ DCGM_ST_GENERIC_ERROR an internal error has occurred
- ▶ DCGM_ST_GROUP_INCOMPATIBLE if groupId refers to a group of non-homogeneous GPUs. This is currently not allowed.

Description

Inform the action manager to perform a manual validation of a group of GPUs on the system

`dcgmReturn_t dcgmRunDiagnostic (dcgmHandle_t pDcgmHandle,
dcgmGpuGrp_t groupId, dcgmDiagnosticLevel_t diagLevel,
dcgmDiagResponse_t *diagResponse)`

Parameters**pDcgmHandle**

IN: DCGM Handle

groupId

IN: Group ID representing collection of one or more GPUs. Look at [dcgmGroupCreate](#) for details on creating the group. Alternatively, pass in the group id as DCGM_GROUP_ALL_GPUS to perform operation on all the GPUs.

diagLevel

IN: Diagnostic level to run

diagResponse

IN/OUT: Result of running the DCGM diagnostic. .version should be set to [dcgmDiagResponse_version](#) before this call.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_NOT_SUPPORTED if running the diagnostic is not supported. This is usually due to the Tesla recommended driver not being installed on the system.
- ▶ DCGM_ST_BADPARAM if a provided parameter is invalid or missing
- ▶ DCGM_ST_GENERIC_ERROR an internal error has occurred
- ▶ DCGM_ST_GROUP_INCOMPATIBLE if groupId refers to a group of non-homogeneous GPUs. This is currently not allowed.
- ▶ DCGM_ST_VER_MISMATCH if .version is not set or is invalid.

Description

Run a diagnostic on a group of GPUs

2.9. Topology

dcmReturn_t dcgmGetDeviceTopology
(dcgmHandle_t pDcgmHandle, unsigned int gpuld,
dcgmDeviceTopology_t *pDcgmDeviceTopology)

Parameters**pDcgmHandle**

IN: DCGM Handle

gpuld

IN: GPU Id corresponding to which topology information should be fetched

pDcgmDeviceTopology

IN/OUT: Topology information corresponding to gpuld. pDcgmDeviceTopology->version must be set to dcgmDeviceTopology_version before this call.

Returns

- ▶ DCGM_ST_OK if the call was successful.
- ▶ DCGM_ST_BADPARAM if gpuld or pDcgmDeviceTopology were not valid.
- ▶ DCGM_ST_VER_MISMATCH if pDcgmDeviceTopology->version was not set to dcgmDeviceTopology_version.

Description

Gets device topology corresponding to the gpuld.

dcgmReturn_t dcgmGetGroupTopology (dcgmHandle_t pDcgmHandle, dcgmGpuGrp_t groupId, dcgmGroupTopology_t *pDcgmGroupTopology)

Parameters

pDcgmHandle

IN: DCGM Handle

groupId

IN: GroupId corresponding to which topology information should be fetched

pDcgmGroupTopology

IN/OUT: Topology information corresponding to groupId. pDcgmgroupTopology->version must be set to dcgmGroupTopology_version.

Returns

- ▶ DCGM_ST_OK if the call was successful.
- ▶ DCGM_ST_BADPARAM if groupId or pDcgmGroupTopology were not valid.
- ▶ DCGM_ST_VER_MISMATCH if pDcgmgroupTopology->version was not set to dcgmGroupTopology_version.

Description

Gets group topology corresponding to the groupId.

2.10. Metadata

This chapter describes the methods that query for DCGM metadata.

dcgmReturn_t dcgmlntrospectToggleState (dcgmHandle_t pDcgmHandle, dcgmlntrospectState_t enabledState)

Parameters

pDcgmHandle

IN: DCGM Handle

enabledState

IN: The state to set gathering of introspection data to

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM enabledState is an invalid state for metadata gathering

Description

Toggle the state of introspection metadata gathering in DCGM. Metadata gathering will increase the memory usage of DCGM so that it can store the metadata it gathers.

```
dcgmReturn_t dcgmIntrospectGetFieldsMemoryUsage
(dcgmHandle_t pDcgmHandle, dcgmIntrospectContext_t
 *context, dcgmIntrospectFullMemory_t *memoryInfo, int
waitIfNoData)
```

Parameters**pDcgmHandle**

IN: DCGM Handle

context

IN: see [dcgmIntrospectContext_t](#). This identifies the level of fields to do introspection for (ex: all fields, field groups) context->version must be set to dcgmIntrospectContext_version prior to this call.

memoryInfo

IN/OUT: see [dcgmIntrospectFullMemory_t](#). memoryInfo->version must be set to dcgmIntrospectFullMemory_version prior to this call.

waitIfNoData

IN: if no metadata has been gathered, should this call block until data has been gathered (1), or should this call just return DCGM_ST_NO_DATA (0).

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_NOT_CONFIGURED if metadata gathering state is DCGM_INTROSPECT_STATE_DISABLED
- ▶ DCGM_ST_NO_DATA if waitIfNoData is false and metadata has not been gathered yet
- ▶ DCGM_ST_VER_MISMATCH if context->version or memoryInfo->version is 0 or invalid.

Description

Get the current amount of memory used to store the given field collection.

```
dcgmReturn_t
dcgmlIntrospectGetHostengineMemoryUsage
(dcgmHandle_t pDcgmHandle, dcgmlIntrospectMemory_t
*memoryInfo, int waitIfNoData)
```

Parameters

pDcgmHandle

IN: DCGM Handle

memoryInfo

IN/OUT: see [dcgmlIntrospectMemory_t](#). memoryInfo->version must be set to dcgmlIntrospectMemory_version prior to this call.

waitIfNoData

IN: if no metadata is gathered wait till this occurs (!0) or return

DCGM_ST_NO_DATA (0)

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_NOT_CONFIGURED if metadata gathering state is DCGM_INTROSPECT_STATE_DISABLED
- ▶ DCGM_ST_NO_DATA if waitIfNoData is false and metadata has not been gathered yet
- ▶ DCGM_ST_VER_MISMATCH if memoryInfo->version is 0 or invalid.

Description

Retrieve the total amount of memory that the hostengine process is currently using. This measurement represents both the resident set size (what is currently in RAM) and the swapped memory that belongs to the process.

```
dcgmReturn_t dcgmlIntrospectGetFieldsExecTime
(dcgmHandle_t pDcgmHandle, dcgmlIntrospectContext_t
*context, dcgmlIntrospectFullFieldsExecTime_t
*execTime, int waitIfNoData)
```

Parameters

pDcgmHandle

IN: DCGM Handle

context

IN: see `dcmIntrospectContext_t`. This identifies the level of fields to do introspection for (ex: all fields, field group) context->version must be set to `dcmIntrospectContext_version` prior to this call.

execTime

IN/OUT: see `dcmIntrospectFullFieldsExecTime_t`. execTime->version must be set to `dcmIntrospectFullFieldsExecTime_version` prior to this call.

waitIfNoData

IN: if no metadata is gathered, wait until data has been gathered (1) or return `DCGM_ST_NO_DATA` (0)

Returns

- ▶ `DCGM_ST_OK` if the call was successful
- ▶ `DCGM_ST_NOT_CONFIGURED` if metadata gathering state is `DCGM_INTROSPECT_STATE_DISABLED`
- ▶ `DCGM_ST_NO_DATA` if `waitIfNoData` is false and metadata has not been gathered yet
- ▶ `DCGM_ST_VER_MISMATCH` if `context->version` or `execTime->version` is 0 or invalid.

Description

Get introspection info relating to execution time needed to update the fields identified by context.

`dcmReturn_t dcmIntrospectUpdateAll (dcgmHandle_t pDcmHandle, int waitForUpdate)`

Parameters**pDcmHandle**

IN: DCGM Handle

waitForUpdate

IN: Whether or not to wait for the update loop to complete before returning to the caller

Returns

- ▶ `DCGM_ST_OK` if the call was successful
- ▶ `DCGM_ST_BADPARAM` if `waitForUpdate` is invalid

Description

This method is used to manually tell the the introspection module to update all DCGM introspection data. This is normally performed automatically on an interval of 1 second.

2.11. Topology

This chapter describes the methods that query for DCGM topology information.

```
dcgmReturn_t dcgmSelectGpusByTopology
(dcgmHandle_t pDcgmHandle, uint64_t inputGpuIds,
uint32_t numGpus, uint64_t *outputGpuIds, uint64_t hintFlags)
```

Parameters

pDcgmHandle

IN: DCGM Handle

inputGpuIds

IN: a bitmask of which GPUs DCGM should consider. If some of the GPUs on the system are already in use, they shouldn't be included in the bitmask. 0 means that all of the GPUs in the system should be considered.

numGpus

IN: the number of GPUs that are desired from inputGpuIds. If this number is greater than the number of healthy GPUs in inputGpuIds, then less than numGpus gpus will be specified in outputGpuIds.

outputGpuIds

OUT: a bitmask of numGpus or fewer GPUs from inputGpuIds that represent the best placement available from inputGpuIds.

hintFlags

IN: a bitmask of DCGM_TOPO_HINT_F_ defines of hints that should be taken into account when assigning outputGpuIds.

Returns

- ▶ DCGM_ST_OK if the call was successful

Description

Get the best group of gpus from the specified bitmask according to topological proximity: cpuAffinity, NUMA node, and NVLink.

`dcmReturn_t dcgmGetFieldSummary (dcgmHandle_t pDcgmHandle, dcgmFieldSummaryRequest_t *request)`

Parameters

pDcgmHandle

IN: DCGM Handle

request

IN / OUT: a pointer to the struct detailing the request and containing the response

Description

Get a summary of the values for a field id over a period of time.

2.12. Modules

This chapter describes the methods that query and configure DCGM modules.

`dcmReturn_t dcgmModuleBlacklist (dcgmHandle_t pDcgmHandle, dcgmModuleId_t moduleId)`

Parameters

pDcgmHandle

IN: DCGM Handle

moduleId

IN: ID of the module to blacklist. Use [dcgmModuleGetStatuses](#) to get a list of valid module IDs.

Returns

- ▶ DCGM_ST_OK if the module has been blacklisted.
- ▶ DCGM_ST_IN_USE if the module has already been loaded and cannot be blacklisted.
- ▶ DCGM_ST_BADPARAM if a parameter is missing or bad.

Description

Set a module to be blacklisted. This module will be prevented from being loaded if it hasn't been loaded already. Modules are lazy-loaded as they are used by DCGM APIs, so it's important to call this API soon after the host engine has been started. You can also pass --blacklist-modules to the nv-hostengine binary to make sure modules get blacklisted immediately after the host engine starts up.

dcmReturn_t dcgmModuleGetStatuses (dcgmHandle_t pDcgmHandle, dcgmModuleGetStatuses_t *moduleStatuses)

Parameters

pDcgmHandle

IN: DCGM Handle

moduleStatuses

OUT: Module statuses. .version should be set to dcgmModuleStatuses_version upon calling.

Returns

- ▶ DCGM_ST_OK if the request succeeds.
- ▶ DCGM_ST_BADPARAM if a parameter is missing or bad.

Description

Get the status of all of the DCGM modules.

2.13. Enums and Macros

enum dcgmOperationMode_t

Operation mode for DCGM

DCGM can run in auto-mode where it runs additional threads in the background to collect any metrics of interest and auto manages any operations needed for policy management.

DCGM can also operate in manual-mode where its execution is controlled by the user. In this mode, the user has to periodically call APIs such as [dcgmPolicyTrigger](#) and [dcgmUpdateAllFields](#) which tells DCGM to wake up and perform data collection and operations needed for policy management.

Values

DCGM_OPERATION_MODE_AUTO = 1
DCGM_OPERATION_MODE_MANUAL = 2

enum dcgmOrder_t

When more than one value is returned from a query, which order should it be returned in?

Values

DCGM_ORDER_ASCENDING = 1

Data with earliest (lowest) timestamps returned first.

DCGM_ORDER_DESCENDING = 2

Data with latest (highest) timestamps returned first.

enum dcgmReturn_t

Return values for DCGM API calls.

Values

DCGM_ST_OK = 0

Success.

DCGM_ST_BADPARAM = -1

A bad parameter was passed to a function.

DCGM_ST_GENERIC_ERROR = -3

A generic, unspecified error.

DCGM_ST_MEMORY = -4

An out of memory error occurred.

DCGM_ST_NOT_CONFIGURED = -5

Setting not configured.

DCGM_ST_NOT_SUPPORTED = -6

Feature not supported.

DCGM_ST_INIT_ERROR = -7

DCGM Init error.

DCGM_ST_NVML_ERROR = -8

When NVML returns error.

DCGM_ST_PENDING = -9

Object is in pending state of something else.

DCGM_ST_UNINITIALIZED = -10

Object is in undefined state.

DCGM_ST_TIMEOUT = -11

Requested operation timed out.

DCGM_ST_VER_MISMATCH = -12

Version mismatch between received and understood API.

DCGM_ST_UNKNOWN_FIELD = -13

Unknown field id.

DCGM_ST_NO_DATA = -14

No data is available.

DCGM_ST_STALE_DATA = -15

Data is considered stale.

DCGM_ST_NOT_WATCHED = -16

The given field id is not being updated by the cache manager.

DCGM_ST_NO_PERMISSION = -17

Do not have permission to perform the desired action.

DCGM_ST_GPU_IS_LOST = -18

GPU is no longer reachable.

DCGM_ST_RESET_REQUIRED = -19

GPU requires a reset.

DCGM_ST_FUNCTION_NOT_FOUND = -20

The function that was requested was not found (bindings only error).

DCGM_ST_CONNECTION_NOT_VALID = -21

The connection to the host engine is not valid any longer.

DCGM_ST_GPU_NOT_SUPPORTED = -22

This GPU is not supported by DCGM.

DCGM_ST_GROUP_INCOMPATIBLE = -23

The GPUs of the provided group are not compatible with each other for the requested operation.

DCGM_ST_MAX_LIMIT = -24

Max limit reached for the object.

DCGM_ST_LIBRARY_NOT_FOUND = -25

DCGM library could not be found.

DCGM_ST_DUPLICATE_KEY = -26

Duplicate key passed to a function.

DCGM_ST_GPU_IN_SYNC_BOOST_GROUP = -27

GPU is already a part of a sync boost group.

DCGM_ST_GPU_NOT_IN_SYNC_BOOST_GROUP = -28

GPU is not a part of a sync boost group.

DCGM_ST_REQUIRES_ROOT = -29

This operation cannot be performed when the host engine is running as non-root.

DCGM_ST_NVVS_ERROR = -30

DCGM GPU Diagnostic was successfully executed, but reported an error.

DCGM_ST_INSUFFICIENT_SIZE = -31

An input argument is not large enough.

DCGM_ST_FIELD_UNSUPPORTED_BY_API = -32

The given field ID is not supported by the API being called.

DCGM_ST_MODULE_NOT_LOADED = -33

This request is serviced by a module of DCGM that is not currently loaded.

DCGM_ST_IN_USE = -34

The requested operation could not be completed because the affected resource is in use.

enum dcgmGroupType_t

Type of GPU groups

Values**DCGM_GROUP_DEFAULT = 0**

All the GPUs on the node are added to the group.

DCGM_GROUP_EMPTY = 1

Creates an empty group.

DCGM_GROUP_DEFAULT_NVSWITCHES = 2

All NvSwitches of the node are added to the group.

enum dcgmConfigType_t

Represents the type of configuration to be fetched from the GPUs

Values**DCGM_CONFIG_TARGET_STATE = 0**

The target configuration values to be applied.

DCGM_CONFIG_CURRENT_STATE = 1

The current configuration state.

enum dcgmConfigPowerLimitType_t

Represents the power cap for each member of the group.

Values**DCGM_CONFIG_POWER_CAP_INDIVIDUAL = 0**

Represents the power cap to be applied for each member of the group.

DCGM_CONFIG_POWER_BUDGET_GROUP = 1

Represents the power budget for the entire group.

#define DCGM_INT32_BLANK 0x7fffffff0

Represents value of the field which can be returned by Host Engine in case the operation is not successful Base value for 32 bits integer blank. can be used as an unspecified blank

```
#define DCGM_INT64_BLANK 0x7fffffffffffff0
```

Base value for 64 bits integer blank. can be used as an unspecified blank

```
#define DCGM_FP64_BLANK 140737488355328.0
```

Base value for double blank. 2^{47} . FP 64 has 52 bits of mantissa, so 47 bits can still increment by 1 and represent each value from 0-15

```
#define DCGM_STR_BLANK "<<<NULL>>>"
```

Base value for string blank.

```
#define DCGM_INT32_NOT_FOUND (DCGM_INT32_BLANK +1)
```

Represents an error where INT32 data was not found

```
#define DCGM_INT64_NOT_FOUND (DCGM_INT64_BLANK +1)
```

Represents an error where INT64 data was not found

```
#define DCGM_FP64_NOT_FOUND (DCGM_FP64_BLANK +1.0)
```

Represents an error where FP64 data was not found

```
#define DCGM_STR_NOT_FOUND "<<<NOT_FOUND>>>"
```

Represents an error where STR data was not found

```
#define DCGM_INT32_NOT_SUPPORTED  
(DCGM_INT32_BLANK+2)
```

Represents an error where fetching the INT32 value is not supported

```
#define DCGM_INT64_NOT_SUPPORTED  
(DCGM_INT64_BLANK+2)
```

Represents an error where fetching the INT64 value is not supported

```
#define DCGM_FP64_NOT_SUPPORTED  
(DCGM_FP64_BLANK+2.0)
```

Represents an error where fetching the FP64 value is not supported

```
#define DCGM_STR_NOT_SUPPORTED  
""><<<NOT_SUPPORTED>>>"
```

Represents an error where fetching the STR value is not supported

```
#define DCGM_INT32_NOT_PERMISSIONED  
(DCGM_INT32_BLANK+3)
```

Represents and error where fetching the INT32 value is not allowed with our current credentials

```
#define DCGM_INT64_NOT_PERMISSIONED  
(DCGM_INT64_BLANK+3)
```

Represents and error where fetching the INT64 value is not allowed with our current credentials

```
#define DCGM_FP64_NOT_PERMISSIONED  
(DCGM_FP64_BLANK+3.0)
```

Represents and error where fetching the FP64 value is not allowed with our current credentials

```
#define DCGM_STR_NOT_PERMISSIONED  
""><<<NOT_PERM>>>"
```

Represents and error where fetching the STR value is not allowed with our current credentials

```
#define DCGM_INT32_IS_BLANK (((val) >=  
DCGM_INT32_BLANK) ? 1 : 0)
```

Macro to check if a INT32 value is blank or not

```
#define DCGM_INT64_IS_BLANK (((val) >=
DCGM_INT64_BLANK) ? 1 : 0)
```

Macro to check if a INT64 value is blank or not

```
#define DCGM_FP64_IS_BLANK (((val) >=
DCGM_FP64_BLANK ? 1 : 0))
```

Macro to check if a FP64 value is blank or not

```
#define DCGM_STR_IS_BLANK (val == strstr(val, "<<<")
&& strstr(val, ">>>"))
```

Macro to check if a STR value is blank or not Works on (char *). Looks for <<< at first position and >>> inside string

```
#define DCGM_MAX_NUM_DEVICES 16
```

Max number of GPUs supported by DCGM

```
#define DCGM_NVLINK_MAX_LINKS_PER_GPU 6
```

Number of NvLink links per GPU supported by DCGM This is 6 for Volta and 4 for Pascal

```
#define DCGM_MAX_NUM_SWITCHES 12
```

Max number of NvSwitches supported by DCGM

```
#define DCGM_NVLINK_MAX_LINKS_PER_NVSWITCH 18
```

Number of NvLink links per NvSwitch supported by DCGM

```
#define DCGM_MAX_VGPU_INSTANCES_PER_PGPU 32
```

Maximum number of vGPU instances per physical GPU

```
#define DCGM_MAX_NUM_VGPU_DEVICES
DCGM_MAX_NUM_DEVICES *
DCGM_MAX_VGPU_INSTANCES_PER_PGPU
```

Max number of vGPUs supported on DCGM

#define DCGM_MAX_STR_LENGTH 256

Max length of the DCGM string field

#define DCGM_MAX_CLOCKS 256

Max number of clocks supported for a device

#define DCGM_MAX_NUM_GROUPS 64

Max limit on the number of groups supported by DCGM

#define DCGM_MAX_FBC_SESSIONS 256

Max number of active FBC sessions

#define DCGM_VGPU_NAME_BUFFER_SIZE 64

Represents the size of a buffer that holds a vGPU type Name or vGPU class type or name of process running on vGPU instance.

#define DCGM_GRID_LICENSE_BUFFER_SIZE 128

Represents the size of a buffer that holds a vGPU license string

#define DCGM_CONFIG_COMPUTEMODE_DEFAULT 0

Default compute mode -- multiple contexts per device

#define DCGM_CONFIG_COMPUTEMODE_PROHIBITED 1

Compute-prohibited mode -- no contexts per device

#define

DCGM_CONFIG_COMPUTEMODE_EXCLUSIVE_PROCESS 2

Compute-exclusive-process mode -- only one context per device, usable from multiple threads at a time

#define DCGM_HE_PORT_NUMBER 5555

Default Port Number for DCGM Host Engine

```
#define MAKE_DCGM_VERSION (unsigned int)  
(sizeof(typeName) | ((ver)<<24))
```

Creates a unique version number for each struct

```
#define DCGM_GROUP_ALL_GPUS 0x7fffffff
```

Identifies for special DCGM groups

```
#define DCGM_GROUP_MAX_ENTITIES 64
```

Maximum number of entities per entity group

2.14. Structure definitions

```
struct dcgmConnectV2Params_v1
struct dcgmConnectV2Params_v2
struct dcgmGroupInfo_v1
struct dcgmGroupEntityPair_t
struct dcgmGroupInfo_v2
struct dcgmFieldGroupInfo_v1
struct dcgmErrorInfo_t
struct dcgmClockSet_v1
struct dcgmDeviceSupportedClockSets_v1
struct dcgmDevicePidAccountingStats_v1
struct dcgmDeviceThermals_v1
struct dcgmDevicePowerLimits_v1
struct dcgmDeviceIdentifiers_v1
struct dcgmDeviceMemoryUsage_v1
struct dcgmDeviceVgpuUtilInfo_v1
struct dcgmDeviceEncStats_v1
struct dcgmDeviceFbcStats_v1
struct dcgmDeviceFbcSessionInfo_v1
```

```
struct dcgmDeviceFbcSessions_v1
struct dcgmDeviceVgpuEncSessions_v1
struct dcgmDeviceVgpuProcessUtilInfo_v1
struct dcgmDeviceVgpuls_v1
struct dcgmDeviceVgpuTypeInfo_v1
struct dcgmDeviceAttributes_v1
struct dcgmVgpuDeviceAttributes_v5
struct dcgmVgpuInstanceAttributes_v1
struct dcgmConfigPerfStateSettings_t
struct dcgmConfigPowerLimit_t
struct dcgmConfig_v1
struct dcgmVgpuConfig_v1
struct dcgmPolicyConditionParms_t
struct dcgmPolicyViolationNotify_t
struct dcgmPolicy_v1
struct dcgmPolicyConditionDbe_t
struct dcgmPolicyConditionPci_t
struct dcgmPolicyConditionMpr_t
```

```
struct dcgmPolicyConditionThermal_t
struct dcgmPolicyConditionPower_t
struct dcgmPolicyConditionNvlink_t
struct dcgmPolicyConditionXID_t
struct dcgmPolicyCallbackResponse_v1
struct dcgmFieldValue_v1
struct dcgmFieldValue_v2
struct dcgmStatSummaryInt64_t
struct dcgmStatSummaryInt32_t
struct dcgmStatSummaryFp64_t
struct dcgmHealthResponse_v1
struct dcgmHealthResponse_v2
struct dcgmProcessUtilInfo_t
struct dcgmProcessUtilSample_t
struct dcgmPidSingleInfo_t
struct dcgmPidInfo_v1
struct dcgmGpuUsageInfo_t
struct dcgmJobInfo_v2
```

```
struct dcgmRunningProcess_v1
struct dcgmDiagResponsePerGpu_t
struct dcgmDiagResponse_v3
struct dcgmDeviceTopology_v1
struct dcgmGroupTopology_v1
struct dcgmlintrospectContext_v1
struct dcgmlintrospectFieldsExecTime_v1
struct dcgmlintrospectFullFieldsExecTime_v1
struct dcgmlintrospectMemory_v1
struct dcgmlintrospectFullMemory_v1
struct dcgmlintrospectCpuUtil_v1
struct dcgmNvLinkGpuLinkStatus_t
struct dcgmNvLinkNvSwitchLinkStatus_t
struct dcgmNvLinkStatus_v1
struct dcgmModuleGetStatusesModule_t
enum dcgmPolicyCondition_t
```

Enumeration for policy conditions. When used as part of dcgmPolicy_t these have corresponding parameters to allow them to be switched on/off or set specific violation thresholds

Values**DCGM_POLICY_COND_DBE = 0x1**

Double bit errors -- boolean in dcgmPolicyConditionParms_t.

DCGM_POLICY_COND_PCI = 0x2

PCI events/errors -- boolean in dcgmPolicyConditionParms_t.

DCGM_POLICY_COND_MAX_PAGES_RETired = 0x4

Maximum number of retired pages -- number required in dcgmPolicyConditionParms_t.

DCGM_POLICY_COND_THERMAL = 0x8

Thermal violation -- number required in dcgmPolicyConditionParms_t.

DCGM_POLICY_COND_POWER = 0x10

Power violation -- number required in dcgmPolicyConditionParms_t.

DCGM_POLICY_COND_NVLINK = 0x20

NVLINK errors -- boolean in dcgmPolicyConditionParms_t.

DCGM_POLICY_COND_XID = 0x40

XID errors -- number required in dcgmPolicyConditionParms_t.

enum dcgmPolicyMode_t

Enumeration for policy modes

Values**DCGM_POLICY_MODE_AUTOMATED = 0**

automatic mode

DCGM_POLICY_MODE_MANUAL = 1

manual mode

enum dcgmPolicyIsolation_t

Enumeration for policy isolation modes

Values**DCGM_POLICY_ISOLATION_NONE = 0**

no isolation of GPUs on error

enum dcgmPolicyAction_t

Enumeration for policy actions

Values**DCGM_POLICY_ACTION_NONE = 0**

no action

DCGM_POLICY_ACTION_GPURESET = 1
perform a GPU reset on violation

enum dcgmPolicyValidation_t

Enumeration for policy validation actions

Values

DCGM_POLICY_VALID_NONE = 0
no validation after an action is performed
DCGM_POLICY_VALID_SV_SHORT = 1
run a short System Validation on the system after failure
DCGM_POLICY_VALID_SV_MED = 2
run a medium System Validation test after failure
DCGM_POLICY_VALID_SV_LONG = 3
run a extensive System Validation test after failure

enum dcgmPolicyFailureResp_t

Enumeration for policy failure responses

Values

DCGM_POLICY_FAILURE_NONE = 0
on failure of validation perform no action

enum dcgmHealthSystems_t

Systems structure used to enable or disable health watch systems

Values

DCGM_HEALTH_WATCH_PCIE = 0x1
PCIe system watches (must have 1m of data before query).
DCGM_HEALTH_WATCH_NVLINK = 0x2
NVLINK system watches.
DCGM_HEALTH_WATCH_PMU = 0x4
Power management unit watches.
DCGM_HEALTH_WATCH_MCU = 0x8
Microcontroller unit watches.
DCGM_HEALTH_WATCH_MEM = 0x10
Memory watches.
DCGM_HEALTH_WATCH_SM = 0x20
Streaming multiprocessor watches.

DCGM_HEALTH_WATCH_INFOROM = 0x40

Inforom watches.

DCGM_HEALTH_WATCH_THERMAL = 0x80

Temperature watches (must have 1m of data before query).

DCGM_HEALTH_WATCH_POWER = 0x100

Power watches (must have 1m of data before query).

DCGM_HEALTH_WATCH_DRIVER = 0x200

Driver-related watches.

DCGM_HEALTH_WATCH_NVSWITCH_NONFATAL = 0x400

Non-fatal errors in NvSwitch.

DCGM_HEALTH_WATCH_NVSWITCH_FATAL = 0x800

Fatal errors in NvSwitch.

DCGM_HEALTH_WATCH_ALL = 0xFFFFFFFF

All watches enabled.

enum dcgmHealthWatchResults_t

Health Watch test results

Values

DCGM_HEALTH_RESULT_PASS = 0

All results within this system are reporting normal.

DCGM_HEALTH_RESULT_WARN = 10

A warning has been issued, refer to the response for more information.

DCGM_HEALTH_RESULT_FAIL = 20

A failure has been issued, refer to the response for more information.

enum dcgmDiagnosticLevel_t

Enumeration for diagnostic levels

Values

DCGM_DIAG_LVL_INVALID = 0

Uninitialized.

DCGM_DIAG_LVL_SHORT = 10

run a very basic health check on the system

DCGM_DIAG_LVL_MED = 20

run a medium-length diagnostic (a few minutes)

DCGM_DIAG_LVL_LONG = 30

run a extensive diagnostic (several minutes)

enum dcgmDiagResult_t

Diagnostic test results

Values

DCGM_DIAG_RESULT_PASS = 0

This test passed as diagnostics.

DCGM_DIAG_RESULT_SKIP = 1

This test was skipped.

DCGM_DIAG_RESULT_WARN = 2

This test passed with warnings.

DCGM_DIAG_RESULT_FAIL = 3

This test failed the diagnostics.

DCGM_DIAG_RESULT_NOT_RUN = 4

This test wasn't executed.

enum dcgmPerGpuTestIndices_t

Diagnostic per gpu tests - fixed indices for `dcgmDiagResponsePerGpu_t.results[]`

Values

DCGM_MEMORY_INDEX = 0

Memory test index.

DCGM_DIAGNOSTIC_INDEX = 1

Diagnostic test index.

DCGM_PCI_INDEX = 2

PCIe test index.

DCGM_SM_PERF_INDEX = 3

SM Stress test index.

DCGM_TARGETED_PERF_INDEX = 4

Targeted Stress test index.

DCGM_TARGETED_POWER_INDEX = 5

Targeted Power test index.

DCGM_MEMORY_BANDWIDTH_INDEX = 6

Memory bandwidth test index.

enum dcgmGpuTopologyLevel_t

Represents level relationships within a system between two GPUs The enums are spaced to allow for future relationships. These match the definitions in nvml.h

Values**DCGM_TOPOLOGY_BOARD** = 0x1

multi-GPU board

DCGM_TOPOLOGY_SINGLE = 0x2

all devices that only need traverse a single PCIe switch

DCGM_TOPOLOGY_MULTIPLE = 0x4

all devices that need not traverse a host bridge

DCGM_TOPOLOGY_HOSTBRIDGE = 0x8

all devices that are connected to the same host bridge

DCGM_TOPOLOGY_CPU = 0x10

all devices that are connected to the same CPU but possibly multiple host bridges

DCGM_TOPOLOGY_SYSTEM = 0x20

all devices in the system

DCGM_TOPOLOGY_NVLINK1 = 0x0100

GPUs connected via a single NVLINK link.

DCGM_TOPOLOGY_NVLINK2 = 0x0200

GPUs connected via two NVLINK links.

DCGM_TOPOLOGY_NVLINK3 = 0x0400

GPUs connected via three NVLINK links.

DCGM_TOPOLOGY_NVLINK4 = 0x0800

GPUs connected via four NVLINK links.

DCGM_TOPOLOGY_NVLINK5 = 0x1000

GPUs connected via five NVLINK links.

DCGM_TOPOLOGY_NVLINK6 = 0x2000

GPUs connected via six NVLINK links.

enum dcgmIntrospectLevel_t

Identifies a level to retrieve field introspection info for

Values**DCGM_INTROSPECT_LVL_INVALID** = 0

Invalid value.

DCGM_INTROSPECT_LVL_FIELD = 1

Introspection data is grouped by field ID.

DCGM_INTROSPECT_LVL_FIELD_GROUP = 2

Introspection data is grouped by field group.

DCGM_INTROSPECT_LVL_ALL_FIELDS

Introspection data is aggregated for all fields.

enum dcgmIntrospectState_t

State of DCGM metadata gathering. If it is set to DISABLED then "Metadata" API calls to DCGM are not supported.

Values

DCGM_INTROSPECT_STATE_DISABLED = 0
DCGM_INTROSPECT_STATE_ENABLED = 1

enum dcgmGpuNvLinkErrorType_t

Identifies a GPU NVLink error type returned by
DCGM_FI_DEV_GPU_NVLINK_ERRORS

Values

DCGM_GPU_NVLINK_ERROR_RECOVERY_REQUIRED = 1
 NVLink link recovery error occurred.
DCGM_GPU_NVLINK_ERROR_FATAL
 NVLink link fatal error occurred.

enum dcgmNvLinkLinkState_t

NvLink link states

Values

DcgmNvLinkLinkStateNotSupported = 0
 NvLink is unsupported by this GPU (Default for GPUs).
DcgmNvLinkLinkStateDisabled = 1
 NvLink is supported for this link but this link is disabled (Default for NvSwitches).
DcgmNvLinkLinkStateDown = 2
 This NvLink link is down (inactive).
DcgmNvLinkLinkStateUp = 3
 This NvLink link is up (active).

enum dcgmModuleId_t

Module IDs

Values

DcgmModuleIdCore = 0
 Core DCGM - always loaded.

DcgmModuleIdNvSwitch = 1
 NvSwitch Module.

DcgmModuleIdVGPU = 2
 VGPU Module.

DcgmModuleIdIntrospect = 3
 Introspection Module.

DcgmModuleIdHealth = 4
 Health Module.

DcgmModuleIdPolicy = 5
 Policy Module.

DcgmModuleIdConfig = 6
 Config Module.

DcgmModuleIdDiag = 7
 GPU Diagnostic Module.

DcgmModuleIdCount
 Always last. 1 greater than largest value above.

enum dcgmModuleStatus_t

Module Status. Modules are lazy loaded, so they will be in status DcgmModuleStatusNotLoaded until they are used. Once modules are used, they will move to another status.

Values

DcgmModuleStatusNotLoaded = 0
 Module has not been loaded yet.

DcgmModuleStatusBlacklisted = 1
 Module has been blacklisted from being loaded.

DcgmModuleStatusFailed = 2
 Loading the module failed.

DcgmModuleStatusLoaded = 3
 Module has been loaded.

typedef void *dcgmHandle_t

Identifier for DCGM Handle.

typedef void *dcgmGpuGrp_t

Identifier for a group of GPUs. A group can have one or more GPUs.

typedef void *dcgmFieldGrp_t

Identifier for a group of fields.

typedef void *dcgmStatus_t

Identifier for list of status codes.

typedef dcgmConnectV2Params_t

Typedef for [dcgmConnectV2Params_v2](#)

typedef dcgmGroupInfo_t

Typedef for [dcgmGroupInfo_v2](#)

typedef dcgmClockSet_t

Typedef for [dcgmClockSet_v1](#)

typedef dcgmDeviceSupportedClockSets_t

Typedef for [dcgmDeviceSupportedClockSets_v1](#)

typedef dcgmDevicePidAccountingStats_t

Typedef for [dcgmDevicePidAccountingStats_v1](#)

typedef dcgmDeviceThermals_t

Typedef for [dcgmDeviceThermals_v1](#)

typedef dcgmDevicePowerLimits_t

Typedef for [dcgmDevicePowerLimits_v1](#)

typedef dcgmDeviceIdentifiers_t

Typedef for [dcgmDeviceIdentifiers_v1](#)

typedef dcgmDeviceMemoryUsage_t

Typedef for [dcgmDeviceMemoryUsage_v1](#)

typedef dcgmDeviceVgpuUtilInfo_t

Typedef for [dcgmDeviceVgpuUtilInfo_v1](#)

typedef dcgmDeviceEncStats_tTypedef for [dcgmDeviceEncStats_v1](#)**typedef dcgmDeviceFbcStats_t**Typedef for [dcgmDeviceFbcStats_v1](#)**typedef dcgmDeviceFbcSessionInfo_t**Typedef for [dcgmDeviceFbcSessionInfo_v1](#)**typedef dcgmDeviceFbcSessions_t**Typedef for [dcgmDeviceFbcSessions_v1](#)**typedef dcgmDeviceVgpuEncSessions_t**Typedef for [dcgmDeviceVgpuEncSessions_v1](#)**typedef dcgmDeviceVgpuProcessUtilInfo_t**Typedef for [dcgmDeviceVgpuProcessUtilInfo_v1](#)**typedef dcgmDeviceVgpuids_t**Typedef for [dcgmDeviceVgpuIds_v1](#)**typedef dcgmDeviceVgpuTypeInfo_t**Typedef for [dcgmDeviceVgpuTypeInfo_v1](#)**typedef dcgmDeviceAttributes_t**Typedef for [dcgmDeviceAttributes_v1](#)**typedef dcgmVgpuDeviceAttributes_t**Typedef for [dcgmVgpuDeviceAttributes_v5](#)**typedef dcgmVgpuInstanceAttributes_t**Typedef for [dcgmVgpuInstanceAttributes_v1](#)

typedef dcgmConfig_tTypedef for `dcgmConfig_v1`**typedef dcgmVgpuConfig_t**Typedef for `dcgmVgpuConfig_v1`**typedef (*fpRecvUpdates) (void* userData)**

Represents a callback to receive updates from asynchronous functions. Currently the only implemented callback function is `dcgmPolicyRegister` and the `void *` data will be a pointer to `dcgmPolicyCallbackResponse_t`. Ex. `dcgmPolicyCallbackResponse_t *callbackResponse = (dcgmPolicyCallbackResponse_t *) userData;`

typedef dcgmPolicy_tTypedef for `dcgmPolicy_v1`**typedef dcgmPolicyCallbackResponse_t**Typedef for `dcgmPolicyCallbackResponse_v1`**typedef dcgmFieldValue_t**Typedef for `dcgmFieldValue_v2`**typedef (*dcgmFieldValueEnumeration_f) (unsigned int gpuld, dcgmFieldValue_v1* values, int numValues, void* userData)**

User callback function for processing one or more field updates. This callback will be invoked one or more times per field until all of the expected field values have been enumerated. It is up to the callee to detect when the field id changes

Returns 0 if OK <0 if enumeration should stop. This allows to callee to abort field value enumeration.

**typedef (*dcgmFieldValueEntityEnumeration_f)
(dcgm_field_entity_group_t entityId,**

**dcgm_field_eid_t entityId, dcgmFieldValue_v1* values,
int numValues, void* userData)**

User callback function for processing one or more field updates. This callback will be invoked one or more times per field until all of the expected field values have been enumerated. It is up to the callee to detect when the field id changes

Returns 0 if OK <0 if enumeration should stop. This allows to callee to abort field value enumeration.

typedef dcgmHealthResponse_t

Typedef for [dcgmHealthResponse_v2](#)

typedef dcgmPidInfo_t

Typedef for [dcgmPidInfo_v1](#)

typedef dcgmJobInfo_t

Typedef for [dcgmJobInfo_v2](#)

typedef dcgmRunningProcess_t

Typedef for [dcgmRunningProcess_v1](#)

typedef dcgmDiagResponse_t

Typedef for [dcgmDiagResponse_v3](#)

typedef dcgmDeviceTopology_t

Typedef for [dcgmDeviceTopology_v1](#)

typedef dcgmGroupTopology_t

Typedef for [dcgmGroupTopology_v1](#)

typedef dcgmIntrospectContext_t

Typedef for [dcgmIntrospectContext_v1](#)

typedef dcgmIntrospectFieldsExecTime_t

Typedef for [dcgmIntrospectFieldsExecTime_t](#)

typedef dcgmIntrospectFullFieldsExecTime_t

typedef for `dcgmIntrospectFullFieldsExecTime_v1`

typedef dcgmIntrospectMemory_t

Typedef for `dcgmIntrospectMemory_t`

typedef dcgmIntrospectFullMemory_t

typedef for `dcgmIntrospectFullMemory_v1`

typedef dcgmIntrospectCpuUtil_t

Typedef for `dcgmIntrospectCpuUtil_t`

typedef dcgmRunDiag_v2 dcgmRunDiag_t

Typedef for `dcgmRunDiag_t`

#define dcgmConnectV2Params_version1

MAKE_DCGM_VERSION(dcgmConnectV2Params_v1, 1)

Version 1 for `dcgmConnectV2Params_v1`

#define dcgmConnectV2Params_version2

MAKE_DCGM_VERSION(dcgmConnectV2Params_v2, 2)

Version 2 for `dcgmConnectV2Params_v2`

#define dcgmConnectV2Params_version

dcgmConnectV2Params_version2

Latest version for `dcgmConnectV2Params_t`

#define dcgmGroupInfo_version1

MAKE_DCGM_VERSION(dcgmGroupInfo_v1, 1)

Version 1 for `dcgmGroupInfo_v1`

```
#define dcgmGroupInfo_version2  
MAKE_DCGM_VERSION(dcgmGroupInfo_v2, 2)
```

Version 2 for dcgmGroupInfo_v2

```
#define dcgmGroupInfo_version  
dcgmGroupInfo_version2
```

Latest version for dcgmGroupInfo_t

```
#define DCGM_MAX_NUM_FIELD_GROUPS 64
```

Maximum number of field groups that can exist

```
#define DCGM_MAX_FIELD_IDS_PER_FIELD_GROUP 128
```

Maximum number of field IDs that can be in a single field group

```
#define dcgmFieldGroupInfo_version1  
MAKE_DCGM_VERSION(dcgmFieldGroupInfo_v1, 1)
```

Version 1 for dcgmFieldGroupInfo_v1

```
#define dcgmFieldGroupInfo_version  
dcgmFieldGroupInfo_version1
```

Latest version for dcgmFieldGroupInfo_t

```
#define dcgmAllFieldGroup_version1  
MAKE_DCGM_VERSION(dcgmAllFieldGroup_v1, 1)
```

Version 1 for dcgmAllFieldGroup_v1

```
#define dcgmAllFieldGroup_version  
dcgmAllFieldGroup_version1
```

Latest version for dcgmAllFieldGroup_t

```
#define dcgmClockSet_version1  
MAKE_DCGM_VERSION(dcgmClockSet_v1, 1)
```

Version 1 for dcgmClockSet_v1

```
#define dcgmClockSet_version dcgmClockSet_version1
```

Latest version for `dcgmClockSet_t`

```
#define dcgmDeviceSupportedClockSets_version1
```

```
MAKE_DCGM_VERSION(dcgmDeviceSupportedClockSets_v1,  
1)
```

Version 1 for `dcgmDeviceSupportedClockSets_v1`

```
#define dcgmDeviceSupportedClockSets_version
```

```
dcgmDeviceSupportedClockSets_version1
```

Latest version for `dcgmDeviceSupportedClockSets_t`

```
#define dcgmDevicePidAccountingStats_version1
```

```
MAKE_DCGM_VERSION(dcgmDevicePidAccountingStats_v1,  
1)
```

Version 1 for `dcgmDevicePidAccountingStats_v1`

```
#define dcgmDevicePidAccountingStats_version
```

```
dcgmDevicePidAccountingStats_version1
```

Latest version for `dcgmDevicePidAccountingStats_t`

```
#define dcgmDeviceThermals_version1
```

```
MAKE_DCGM_VERSION(dcgmDeviceThermals_v1, 1)
```

Version 1 for `dcgmDeviceThermals_v1`

```
#define dcgmDeviceThermals_version
```

```
dcgmDeviceThermals_version1
```

Latest version for `dcgmDeviceThermals_t`

```
#define dcgmDevicePowerLimits_version1
```

```
MAKE_DCGM_VERSION(dcgmDevicePowerLimits_v1, 1)
```

Version 1 for `dcgmDevicePowerLimits_v1`

```
#define dcgmDevicePowerLimits_version  
dcgmDevicePowerLimits_version1
```

Latest version for `dcgmDevicePowerLimits_t`

```
#define dcgmDeviceIdentifiers_version1  
MAKE_DCGM_VERSION(dcgmDeviceIdentifiers_v1, 1)
```

Version 1 for `dcgmDeviceIdentifiers_v1`

```
#define dcgmDeviceIdentifiers_version  
dcgmDeviceIdentifiers_version1
```

Latest version for `dcgmDeviceIdentifiers_t`

```
#define dcgmDeviceMemoryUsage_version1  
MAKE_DCGM_VERSION(dcgmDeviceMemoryUsage_v1, 1)
```

Version 1 for `dcgmDeviceMemoryUsage_v1`

```
#define dcgmDeviceMemoryUsage_version  
dcgmDeviceMemoryUsage_version1
```

Latest version for `dcgmDeviceMemoryUsage_t`

```
#define dcgmDeviceVgpuUtilInfo_version1  
MAKE_DCGM_VERSION(dcgmDeviceVgpuUtilInfo_v1, 1)
```

Version 1 for `dcgmDeviceVgpuUtilInfo_v1`

```
#define dcgmDeviceVgpuUtilInfo_version  
dcgmDeviceVgpuUtilInfo_version1
```

Latest version for `dcgmDeviceVgpuUtilInfo_t`

```
#define dcgmDeviceEncStats_version1  
MAKE_DCGM_VERSION(dcgmDeviceEncStats_v1, 1)
```

Version 1 for `dcgmDeviceEncStats_v1`

```
#define dcgmDeviceEncStats_version  
dcgmDeviceEncStats_version1
```

Latest version for `dcgmDeviceEncStats_t`

```
#define dcgmDeviceFbcStats_version1  
MAKE_DCGM_VERSION(dcgmDeviceFbcStats_v1, 1)
```

Version 1 for `dcgmDeviceFbcStats_v1`

```
#define dcgmDeviceFbcStats_version  
dcgmDeviceFbcStats_version1
```

Latest version for `dcgmDeviceEncStats_t`

```
#define dcgmDeviceFbcSessionInfo_version1  
MAKE_DCGM_VERSION(dcgmDeviceFbcSessionInfo_v1, 1)
```

Version 1 for `dcgmDeviceFbcSessionInfo_v1`

```
#define dcgmDeviceFbcSessionInfo_version  
dcgmDeviceFbcSessionInfo_version1
```

Latest version for `dcgmDeviceFbcSessionInfo_t`

```
#define dcgmDeviceFbcSessions_version1  
MAKE_DCGM_VERSION(dcgmDeviceFbcSessions_v1, 1)
```

Version 1 for `dcgmDeviceFbcSessions_v1`

```
#define dcgmDeviceFbcSessions_version  
dcgmDeviceFbcSessions_version1
```

Latest version for `dcgmDeviceFbcSessions_t`

```
#define dcgmDeviceVgpuEncSessions_version1  
MAKE_DCGM_VERSION(dcgmDeviceVgpuEncSessions_v1,  
1)
```

Version 1 for `dcgmDeviceVgpuEncSessions_v1`

```
#define dcgmDeviceVgpuEncSessions_version  
dcgmDeviceVgpuEncSessions_version1
```

Latest version for `dcgmDeviceVgpuEncSessions_t`

```
#define dcgmDeviceVgpuProcessUtilInfo_version1  
MAKE_DCGM_VERSION(dcgmDeviceVgpuProcessUtilInfo_v1,  
1)
```

Version 1 for `dcgmDeviceVgpuProcessUtilInfo_v1`

```
#define dcgmDeviceVgpuProcessUtilInfo_version  
dcgmDeviceVgpuProcessUtilInfo_version1
```

Latest version for `dcgmDeviceVgpuProcessUtilInfo_t`

```
#define dcgmDeviceVgpuids_version1  
MAKE_DCGM_VERSION(dcgmDeviceVgpuids_v1, 1)
```

Version 1 for `dcgmDeviceVgpuIds_v1`

```
#define dcgmDeviceVgpuids_version  
dcgmDeviceVgpuids_version1
```

Latest version for `dcgmDeviceVgpuIds_t`

```
#define dcgmDeviceVgpuTypeInfo_version1  
MAKE_DCGM_VERSION(dcgmDeviceVgpuTypeInfo_v1, 1)
```

Version 1 for `dcgmDeviceVgpuTypeInfo_v1`

```
#define dcgmDeviceVgpuTypeInfo_version  
dcgmDeviceVgpuTypeInfo_version1
```

Latest version for `dcgmDeviceVgpuTypeInfo_t`

```
#define dcgmDeviceAttributes_version1  
MAKE_DCGM_VERSION(dcgmDeviceAttributes_v1, 1)
```

Version 1 for `dcgmDeviceAttributes_v1`

```
#define dcgmDeviceAttributes_version  
dcgmDeviceAttributes_version1
```

Latest version for `dcgmDeviceAttributes_t`

```
#define DCGM_MAX_VGPU_TYPES_PER_PGPU 22
```

Maximum number of vGPU types per physical GPU

```
#define dcgmVgpuDeviceAttributes_version5  
MAKE_DCGM_VERSION(dcgmVgpuDeviceAttributes_v5, 1)
```

Version 5 for `dcgmVgpuDeviceAttributes_v5`

```
#define dcgmVgpuDeviceAttributes_version  
dcgmVgpuDeviceAttributes_version5
```

Latest version for `dcgmVgpuDeviceAttributes_t`

```
#define DCGM_DEVICE_UUID_BUFFER_SIZE 80
```

Represents the size of a buffer that holds string related to attributes specific to vGPU instance

```
#define dcgmVgpuInstanceAttributes_version1  
MAKE_DCGM_VERSION(dcgmVgpuInstanceAttributes_v1,  
1)
```

Version 1 for `dcgmVgpuInstanceAttributes_v1`

```
#define dcgmVgpuInstanceAttributes_version  
dcgmVgpuInstanceAttributes_version1
```

Latest version for `dcgmVgpuInstanceAttributes_t`

```
#define dcgmConfig_version1  
MAKE_DCGM_VERSION(dcgmConfig_v1, 1)
```

Version 1 for `dcgmConfig_v1`

```
#define dcgmConfig_version dcgmConfig_version1
```

Latest version for `dcgmConfig_t`

```
#define dcgmVgpuConfig_version1
```

```
MAKE_DCGM_VERSION(dcgmVgpuConfig_v1, 1)
```

Version 1 for `dcgmVgpuConfig_v1`

```
#define dcgmVgpuConfig_version
```

```
dcgmVgpuConfig_version1
```

Latest version for `dcgmVgpuConfig_t`

```
#define dcgmPolicy_version1
```

```
MAKE_DCGM_VERSION(dcgmPolicy_v1, 1)
```

Version 1 for `dcgmPolicy_v1`

```
#define dcgmPolicy_version dcgmPolicy_version1
```

Latest version for `dcgmPolicy_t`

```
#define dcgmPolicyCallbackResponse_version1
```

```
MAKE_DCGM_VERSION(dcgmPolicyCallbackResponse_v1,  
1)
```

Version 1 for `dcgmPolicyCallbackResponse_v1`

```
#define dcgmPolicyCallbackResponse_version
```

```
dcgmPolicyCallbackResponse_version1
```

Latest version for `dcgmPolicyCallbackResponse_t`

```
#define DCGM_MAX_BLOB_LENGTH 4096
```

Set above size of largest blob entry. Currently this is `dcgmDeviceVgpuTypeInfo_v1`.

```
#define dcgmFieldValue_version1
```

```
MAKE_DCGM_VERSION(dcgmFieldValue_v1, 1)
```

Version 1 for `dcgmFieldValue_v1`

```
#define dcgmFieldValue_version2  
MAKE_DCGM_VERSION(dcgmFieldValue_v2, 2)
```

Version 2 for dcgmFieldValue_v2

```
#define dcgmFieldValue_version  
dcgmFieldValue_version2
```

Latest version for dcgmFieldValue_t

```
#define DCGM_FV_FLAG_LIVE_DATA 0x00000001
```

Field value flags used by dcgmEntitiesGetLatestValues

```
#define DCGM_HEALTH_WATCH_COUNT_V1 10
```

For iterating through the dcgmHealthSystems_v1 enum.

```
#define DCGM_HEALTH_WATCH_COUNT_V2 12
```

For iterating through the dcgmHealthSystems_v2 enum.

```
#define dcgmHealthResponse_version1
```

```
MAKE_DCGM_VERSION(dcgmHealthResponse_v1, 1)
```

Version 1 for dcgmHealthResponse_v1

```
#define dcgmHealthResponse_version2
```

```
MAKE_DCGM_VERSION(dcgmHealthResponse_v2, 2)
```

Version 2 for dcgmHealthResponse_v2

```
#define dcgmHealthResponse_version
```

```
dcgmHealthResponse_version2
```

Latest version for dcgmHealthResponse_t

```
#define dcgmPidInfo_version1
```

```
MAKE_DCGM_VERSION(dcgmPidInfo_v1, 1)
```

Version 1 for dcgmPidInfo_v1

```
#define dcgmPidInfo_version dcgmPidInfo_version1
```

Latest version for `dcgmPidInfo_t`

```
#define dcgmJobInfo_version2
```

```
MAKE_DCGM_VERSION(dcgmJobInfo_v2, 2)
```

Version 2 for `dcgmJobInfo_v2`

```
#define dcgmJobInfo_version dcgmJobInfo_version2
```

Latest version for `dcgmJobInfo_t`

```
#define dcgmRunningProcess_version1
```

```
MAKE_DCGM_VERSION(dcgmRunningProcess_v1, 1)
```

Version 1 for `dcgmRunningProcess_v1`

```
#define dcgmRunningProcess_version
```

```
dcgmRunningProcess_version1
```

Latest version for `dcgmRunningProcess_t`

```
#define dcgmDiagResponse_version3
```

```
MAKE_DCGM_VERSION(dcgmDiagResponse_v3, 3)
```

Version 3 for `dcgmDiagResponse_v3`

```
#define dcgmDiagResponse_version
```

```
dcgmDiagResponse_version3
```

Latest version for `dcgmDiagResponse_t`

```
#define dcgmDeviceTopology_version1
```

```
MAKE_DCGM_VERSION(dcgmDeviceTopology_v1, 1)
```

Version 1 for `dcgmDeviceTopology_v1`

```
#define dcgmDeviceTopology_version
```

```
dcgmDeviceTopology_version1
```

Latest version for `dcgmDeviceTopology_t`

```
#define dcgmGroupTopology_version1  
MAKE_DCGM_VERSION(dcgmGroupTopology_v1, 1)
```

Version 1 for dcgmGroupTopology_v1

```
#define dcgmGroupTopology_version  
dcgmGroupTopology_version1
```

Latest version for dcgmGroupTopology_t

```
#define dcgmlIntrospectContext_version1  
MAKE_DCGM_VERSION(dcgmlIntrospectContext_v1, 1)
```

Version 1 for dcgmlIntrospectContext_t

```
#define dcgmlIntrospectContext_version  
dcgmlIntrospectContext_version1
```

Latest version for dcgmlIntrospectContext_t

```
#define dcgmlIntrospectFieldsExecTime_version1  
MAKE_DCGM_VERSION(dcgmlIntrospectFieldsExecTime_v1,  
1)
```

Version 1 for dcgmlIntrospectFieldsExecTime_t

```
#define dcgmlIntrospectFieldsExecTime_version  
dcgmlIntrospectFieldsExecTime_version1
```

Latest version for dcgmlIntrospectFieldsExecTime_t

```
#define dcgmlIntrospectFullFieldsExecTime_version1  
MAKE_DCGM_VERSION(dcgmlIntrospectFullFieldsExecTime_v1,  
1)
```

Version 1 for dcgmlIntrospectFullFieldsExecTime_t

```
#define dcgmlIntrospectFullFieldsExecTime_version  
dcgmlIntrospectFullFieldsExecTime_version1
```

Latest version for `dcgmIntrospectFullFieldsExecTime_t`

```
#define dcgmlIntrospectMemory_version1  
MAKE_DCGM_VERSION(dcgmlIntrospectMemory_v1, 1)
```

Version 1 for `dcgmIntrospectMemory_t`

```
#define dcgmlIntrospectMemory_version  
dcgmlIntrospectMemory_version1
```

Latest version for `dcgmIntrospectMemory_t`

```
#define dcgmlIntrospectFullMemory_version1  
MAKE_DCGM_VERSION(dcgmlIntrospectFullMemory_v1, 1)
```

Version 1 for `dcgmIntrospectFullMemory_t`

```
#define dcgmlIntrospectFullMemory_version  
dcgmlIntrospectFullMemory_version1
```

Latest version for `dcgmIntrospectFullMemory_t`

```
#define dcgmlIntrospectCpuUtil_version1  
MAKE_DCGM_VERSION(dcgmlIntrospectCpuUtil_v1, 1)
```

Version 1 for `dcgmIntrospectCpuUtil_t`

```
#define dcgmlIntrospectCpuUtil_version  
dcgmlIntrospectCpuUtil_version1
```

Latest version for `dcgmIntrospectCpuUtil_t`

```
#define dcgmRunDiag_version1  
MAKE_DCGM_VERSION(dcgmRunDiag_v1, 1)
```

Version 1 for `dcgmRunDiag_t`

```
#define dcgmRunDiag_version2  
MAKE_DCGM_VERSION(dcgmRunDiag_v2, 2)
```

Version 2 for dcgmRunDiag_t

```
#define dcgmRunDiag_version dcgmRunDiag_version2
```

Latest version for dcgmRunDiag_t

```
#define DCGM_GEGE_FLAG_ONLY_SUPPORTED  
0x00000001
```

This mimics the behavior of dcgmGetAllSupportedDevices().

Flags for dcgmGetEntityGroupEntities's flags parameter Only return entities that are supported by DCGM.

```
#define dcgmNvLinkStatus_version1  
MAKE_DCGM_VERSION(dcgmNvLinkStatus_v1, 1)
```

Version 1 of dcgmNvLinkStatus

```
#define DCGM_MODULE_STATUSES_CAPACITY 16
```

This is larger than DcgmModuleIdCount so we can add modules without versioning this request.

```
#define dcgmModuleGetStatuses_version1  
MAKE_DCGM_VERSION(dcgmModuleGetStatuses_v1, 1)
```

Version 1 of dcgmModuleGetStatuses

2.15. Field Types

Field Types are a single byte.

```
#define DCGM_FT_BINARY 'b'
```

Blob of binary data representing a structure

```
#define DCGM_FT_DOUBLE 'd'
```

8-byte double precision

#define DCGM_FT_INT64 'i'

8-byte signed integer

#define DCGM_FT_STRING 's'

Null-terminated ASCII Character string

#define DCGM_FT_TIMESTAMP 't'

8-byte signed integer usec since 1970

2.16. Field Scope

Represents field association with entity scope or global scope.

#define DCGM_FS_GLOBAL 0

Field is global (ex: driver version)

#define DCGM_FS_ENTITY 1

Field is associated with an entity (GPU, VGPU...etc)

#define DCGM_FS_DEVICE DCGM_FS_ENTITY

Field is associated with a device. Deprecated. Use DCGM_FS_ENTITY

**#define DCGM_CUDA_COMPUTE_CAPABILITY_MAJOR
((uint64_t)(x) & 0xFFFF0000)**

DCGM_FI_DEV_CUDA_COMPUTE_CAPABILITY is 16 bits of major version followed by 16 bits of the minor version. These macros separate the two.

**#define DCGM_CLOCKS_THROTTLE_REASON_GPU_IDLE
0x0000000000000001LL**

DCGM_FI_DEV_CLOCK_THROTTLE_REASONS is a bitmap of why the clock is throttled. These macros are masks for relevant throttling, and are a 1:1 map to the NVML reasons documented in nvml.h. The notes for the header are copied below:
Nothing is running on the GPU and the clocks are dropping to Idle state



This limiter may be removed in a later release

```
#define  
DCGM_CLOCKS_THROTTLE_REASON_CLOCKS_SETTING  
0x00000000000000002LL
```

GPU clocks are limited by current setting of applications clocks

```
#define  
DCGM_CLOCKS_THROTTLE_REASON_SW_POWER_CAP  
0x0000000000000004LL
```

SW Power Scaling algorithm is reducing the clocks below requested clocks

```
#define  
DCGM_CLOCKS_THROTTLE_REASON_HW_SLOWDOWN  
0x0000000000000008LL
```

HW Slowdown (reducing the core clocks by a factor of 2 or more) is engaged

This is an indicator of:

- ▶ temperature being too high
- ▶ External Power Brake Assertion is triggered (e.g. by the system power supply)
- ▶ Power draw is too high and Fast Trigger protection is reducing the clocks
- ▶ May be also reported during PState or clock change
- ▶ This behavior may be removed in a later release.

```
#define  
DCGM_CLOCKS_THROTTLE_REASON_SYNC_BOOST  
0x0000000000000010LL
```

Sync Boost

This GPU has been added to a Sync boost group with nvidia-smi or DCGM in order to maximize performance per watt. All GPUs in the sync boost group will boost to the minimum possible clocks across the entire group. Look at the throttle reasons for other GPUs in the system to see why those GPUs are holding this one at lower clocks.

```
#define  
DCGM_CLOCKS_THROTTLE_REASON_SW_THERMAL  
0x000000000000000020LL
```

SW Thermal Slowdown

This is an indicator of one or more of the following:

- ▶ Current GPU temperature above the GPU Max Operating Temperature
- ▶ Current memory temperature above the Memory Max Operating Temperature

```
#define  
DCGM_CLOCKS_THROTTLE_REASON_HW_THERMAL  
0x000000000000000040LL
```

HW Thermal Slowdown (reducing the core clocks by a factor of 2 or more) is engaged

This is an indicator of:

- ▶ temperature being too high

```
#define  
DCGM_CLOCKS_THROTTLE_REASON_HW_POWER_BRAKE  
0x000000000000000080LL
```

HW Power Brake Slowdown (reducing the core clocks by a factor of 2 or more) is engaged

This is an indicator of:

- ▶ External Power Brake Assertion being triggered (e.g. by the system power supply)

```
#define  
DCGM_CLOCKS_THROTTLE_REASON_DISPLAY_CLOCKS  
0x0000000000000000100LL
```

GPU clocks are limited by current setting of Display clocks

2.17. Field Entity

Represents field association with a particular entity

enum dcgm_field_entity_group_t

Enum of possible field entity groups

Values

DCGM_FE_NONE = 0

DCGM_FE_GPU

Field is not associated with an entity. Field scope should be DCGM_FS_GLOBAL
DCGM_FE_VGPU

Field is associated with a GPU entity

DCGM_FE_SWITCH

Field is associated with a VGPU entity

DCGM_FE_COUNT

Field is associated with a Switch entity Number of elements in this enumeration.

Keep this entry last

typedef unsigned int dcgm_field_eid_t

Represents an identifier for an entity within a field entity. For instance, this is the gpuId for DCGM_FE_GPU.

2.18. Field Identifiers

Field Identifiers

DcgmFieldGetById (unsigned short fieldId)

Parameters

fieldId

IN: One of the field IDs (DCGM_FI_?)

Returns

0 On Failure > 0 Pointer to field metadata structure if found.

Description

Get a pointer to the metadata for a field by its field ID. See DCGM_FI_? for a list of field IDs.

DcgmFieldGetByTag (char *tag)

Parameters

tag

IN: Tag for the field of interest

Returns

0 On failure or not found > 0 Pointer to field metadata structure if found

Description

Get a pointer to the metadata for a field by its field tag.

DcgmFieldsInit (void)

Returns

0 On success <0 On error

Description

Initialize the DcgmFields module. Call this once from inside your program

DcgmFieldsTerm (void)

Returns

0 On success <0 On error

Description

Terminates the DcgmFields module. Call this once from inside your program

char *DcgmFieldsGetEntityGroupString (dcgm_field_entity_group_t entityId)

Description

Get the string version of a entityId

Returns Pointer to a string like GPU/NvSwitch..etc Null on error

#define DCGM_FI_UNKNOWN 0

NULL field

#define DCGM_FI_DRIVER_VERSION 1

Driver Version

#define DCGM_FI_DEV_COUNT 4

Number of Devices on the node

#define DCGM_FI_DEV_NAME 50

Name of the GPU device

#define DCGM_FI_DEV_BRAND 51

Device Brand

#define DCGM_FI_DEV_NVML_INDEX 52

NVML index of this GPU

#define DCGM_FI_DEV_SERIAL 53

Device Serial Number

#define DCGM_FI_DEV_UUID 54

UUID corresponding to the device

#define DCGM_FI_DEV_MINOR_NUMBER 55

Device node minor number /dev/nvidia#

#define DCGM_FI_DEV_OEM_INFOROM_VER 56

OEM inforom version

#define DCGM_FI_DEV_PCI_BUSID 57

PCI attributes for the device

#define DCGM_FI_DEV_PCI_COMBINED_ID 58

The combined 16-bit device id and 16-bit vendor id

#define DCGM_FI_DEV_PCI_SUBSYS_ID 59

The 32-bit Sub System Device ID

#define DCGM_FI_GPU_TOPOLOGY_PCI 60

Topology of all GPUs on the system via PCI (static)

#define DCGM_FI_GPU_TOPOLOGY_NVLINK 61

Topology of all GPUs on the system via NVLINK (static)

#define DCGM_FI_GPU_TOPOLOGY_AFFINITY 62

Affinity of all GPUs on the system (static)

#define DCGM_FI_DEV_CUDA_COMPUTE_CAPABILITY 63

Cuda compute capability for the device. The major version is the upper 32 bits and the minor version is the lower 32 bits.

#define DCGM_FI_DEV_COMPUTE_MODE 65

Compute mode for the device

#define DCGM_FI_DEV_CPU_AFFINITY_0 70

Device CPU affinity. part 1/8 = cpus 0 - 63

#define DCGM_FI_DEV_CPU_AFFINITY_1 71

Device CPU affinity. part 1/8 = cpus 64 - 127

#define DCGM_FI_DEV_CPU_AFFINITY_2 72

Device CPU affinity. part 2/8 = cpus 128 - 191

#define DCGM_FI_DEV_CPU_AFFINITY_3 73

Device CPU affinity. part 3/8 = cpus 192 - 255

#define DCGM_FI_DEV_ECC_INFOROM_VER 80

ECC inforom version

#define DCGM_FI_DEV_POWER_INFOROM_VER 81

Power management object inforom version

#define DCGM_FI_DEV_INFOROM_IMAGE_VER 82

Inforom image version

#define DCGM_FI_DEV_INFOROM_CONFIG_CHECK 83

Inforom configuration checksum

#define DCGM_FI_DEV_INFOROM_CONFIG_VALID 84

Reads the infoROM from the flash and verifies the checksums

#define DCGM_FI_DEV_VBIOS_VERSION 85

VBIOS version of the device

#define DCGM_FI_DEV_BAR1_TOTAL 90

Total BAR1 of the GPU in MB

#define DCGM_FI_SYNC_BOOST 91

Sync boost settings on the node

#define DCGM_FI_DEV_BAR1_USED 92

Used BAR1 of the GPU in MB

#define DCGM_FI_DEV_BAR1_FREE 93

Free BAR1 of the GPU in MB

#define DCGM_FI_DEV_SM_CLOCK 100

SM clock for the device

#define DCGM_FI_DEV_MEM_CLOCK 101

Memory clock for the device

#define DCGM_FI_DEV_VIDEO_CLOCK 102

Video encoder/decoder clock for the device

#define DCGM_FI_DEV_APP_SM_CLOCK 110

SM Application clocks

#define DCGM_FI_DEV_APP_MEM_CLOCK 111

Memory Application clocks

#define DCGM_FI_DEV_CLOCK_THROTTLE_REASONS 112

Current clock throttle reasons (bitmask of DCGM_CLOCKS_THROTTLE_REASON_*)

#define DCGM_FI_DEV_AUTOBOOST 120

Auto-boost for the device (1 = enabled. 0 = disabled)

#define DCGM_FI_DEV_SUPPORTED_CLOCKS 130

Supported clocks for the device

#define DCGM_FI_DEV_MEMORY_TEMP 140

Memory temperature for the device

#define DCGM_FI_DEV_GPU_TEMP 150

Current temperature readings for the device, in degrees C

#define DCGM_FI_DEV_POWER_USAGE 155

Power usage for the device in Watts

#define DCGM_FI_DEV_TOTAL_ENERGY_CONSUMPTION 156

Total energy consumption for the GPU in mJ since the driver was last reloaded

#define DCGM_FI_DEV_SLOWDOWN_TEMP 158

Slowdown temperature for the device

#define DCGM_FI_DEV_SHUTDOWN_TEMP 159

Shutdown temperature for the device

#define DCGM_FI_DEV_POWER_MGMT_LIMIT 160

Current Power limit for the device

#define DCGM_FI_DEV_POWER_MGMT_LIMIT_MIN 161

Minimum power management limit for the device

#define DCGM_FI_DEV_POWER_MGMT_LIMIT_MAX 162

Maximum power management limit for the device

#define DCGM_FI_DEV_POWER_MGMT_LIMIT_DEF 163

Default power management limit for the device

#define DCGM_FI_DEV_ENFORCED_POWER_LIMIT 164

Effective power limit that the driver enforces after taking into account all limiters

#define DCGM_FI_DEV_PSTATE 190

Performance state (P-State) 0-15. 0=highest

#define DCGM_FI_DEV_FAN_SPEED 191

Fan speed for the device in percent 0-100

#define DCGM_FI_DEV_PCIE_TX_THROUGHPUT 200

PCIe Tx utilization information

#define DCGM_FI_DEV_PCIE_RX_THROUGHPUT 201

PCIe Rx utilization information

#define DCGM_FI_DEV_PCIE_REPLAY_COUNTER 202

PCIe replay counter

#define DCGM_FI_DEV_GPU_UTIL 203

GPU Utilization

#define DCGM_FI_DEV_MEM_COPY_UTIL 204

Memory Utilization

#define DCGM_FI_DEV_ACCOUNTING_DATA 205

Process accounting stats.

This field is only supported when the host engine is running as root unless you enable accounting ahead of time. Accounting mode can be enabled by running "nvidia-smi -am 1" as root on the same node the host engine is running on.

#define DCGM_FI_DEV_ENC_UTIL 206

Encoder Utilization

#define DCGM_FI_DEV_DEC_UTIL 207

Decoder Utilization

#define DCGM_FI_DEV_MEM_COPY_UTIL_SAMPLES 210

Memory utilization samples

#define DCGM_FI_DEV_GRAPHICS_PIDS 220

Graphics processes running on the GPU.

#define DCGM_FI_DEV_COMPUTE_PIDS 221

Compute processes running on the GPU.

#define DCGM_FI_DEV_XID_ERRORS 230

XID errors. The value is the specific XID error

#define DCGM_FI_DEV_PCIE_MAX_LINK_GEN 235

PCIe Max Link Generation

#define DCGM_FI_DEV_PCIE_MAX_LINK_WIDTH 236

PCIe Max Link Width

#define DCGM_FI_DEV_PCIE_LINK_GEN 237

PCIe Current Link Generation

#define DCGM_FI_DEV_PCIE_LINK_WIDTH 238

PCIe Current Link Width

#define DCGM_FI_DEV_POWER_VIOLATION 240

Power Violation time in usec

#define DCGM_FI_DEV_THERMAL_VIOLATION 241

Thermal Violation time in usec

#define DCGM_FI_DEV_SYNC_BOOST_VIOLATION 242

Sync Boost Violation time in usec

#define DCGM_FI_DEV_BOARD_LIMIT_VIOLATION 243

Board violation limit.

#define DCGM_FI_DEV_LOW_UTIL_VIOLATION 244

Low utilisation violation limit.

#define DCGM_FI_DEV_RELIABILITY_VIOLATION 245

Reliability violation limit.

#define DCGM_FI_DEV_TOTAL_APP_CLOCKS_VIOLATION 246

App clock violation limit.

#define DCGM_FI_DEV_TOTAL_BASE_CLOCKS_VIOLATION 247

Base clock violation limit.

#define DCGM_FI_DEV_FB_TOTAL 250

Total Frame Buffer of the GPU in MB

#define DCGM_FI_DEV_FB_FREE 251

Free Frame Buffer in MB

#define DCGM_FI_DEV_FB_USED 252

Used Frame Buffer in MB

#define DCGM_FI_DEV_ECC_CURRENT 300

Current ECC mode for the device

#define DCGM_FI_DEV_ECC_PENDING 301

Pending ECC mode for the device

#define DCGM_FI_DEV_ECC_SBE_VOL_TOTAL 310

Total single bit volatile ECC errors

#define DCGM_FI_DEV_ECC_DBE_VOL_TOTAL 311

Total double bit volatile ECC errors

#define DCGM_FI_DEV_ECC_SBE_AGG_TOTAL 312

Total single bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_DBE_AGG_TOTAL 313

Total double bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_SBE_VOL_L1 314

L1 cache single bit volatile ECC errors

#define DCGM_FI_DEV_ECC_DBE_VOL_L1 315

L1 cache double bit volatile ECC errors

#define DCGM_FI_DEV_ECC_SBE_VOL_L2 316

L2 cache single bit volatile ECC errors

#define DCGM_FI_DEV_ECC_DBE_VOL_L2 317

L2 cache double bit volatile ECC errors

#define DCGM_FI_DEV_ECC_SBE_VOL_DEV 318

Device memory single bit volatile ECC errors

#define DCGM_FI_DEV_ECC_DBE_VOL_DEV 319

Device memory double bit volatile ECC errors

#define DCGM_FI_DEV_ECC_SBE_VOL_REG 320

Register file single bit volatile ECC errors

#define DCGM_FI_DEV_ECC_DBE_VOL_REG 321

Register file double bit volatile ECC errors

#define DCGM_FI_DEV_ECC_SBE_VOL_TEX 322

Texture memory single bit volatile ECC errors

#define DCGM_FI_DEV_ECC_DBE_VOL_TEX 323

Texture memory double bit volatile ECC errors

#define DCGM_FI_DEV_ECC_SBE_AGG_L1 324

L1 cache single bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_DBE_AGG_L1 325

L1 cache double bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_SBE_AGG_L2 326

L2 cache single bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_DBE_AGG_L2 327

L2 cache double bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_SBE_AGG_DEV 328

Device memory single bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_DBE_AGG_DEV 329

Device memory double bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_SBE_AGG_REG 330

Register File single bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_DBE_AGG_REG 331

Register File double bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_SBE_AGG_TEX 332

Texture memory single bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_ECC_DBE_AGG_TEX 333

Texture memory double bit aggregate (persistent) ECC errors Note: monotonically increasing

#define DCGM_FI_DEV_RETIRED_SBE 390

Number of retired pages because of single bit errors Note: monotonically increasing

#define DCGM_FI_DEV_RETIRED_DBE 391

Number of retired pages because of double bit errors Note: monotonically increasing

#define DCGM_FI_DEV_RETIRED_PENDING 392

Number of pages pending retirement

#define DCGM_FI_DEV_VIRTUAL_MODE 500

Virtualization Mode corresponding to the GPU

#define DCGM_FI_DEV_SUPPORTED_TYPE_INFO 501

Includes Count and Static info of vGPU types supported on a device

#define DCGM_FI_DEV_CREATABLE_VGPU_TYPE_IDS 502

Includes Count and currently Creatable vGPU types on a device

#define DCGM_FI_DEV_VGPU_INSTANCE_IDS 503

Includes Count and currently Active vGPU Instances on a device

#define DCGM_FI_DEV_VGPU_UTILIZATIONS 504

Utilization values for vGPUs running on the device

#define**DCGM_FI_DEV_VGPU_PER_PROCESS_UTILIZATION 505**

Utilization values for processes running within vGPU VMs using the device

#define DCGM_FI_DEV_ENC_STATS 506

Current encoder statistics for a given device

#define DCGM_FI_DEV_FBC_STATS 507

Statistics of current active frame buffer capture sessions on a given device

#define DCGM_FI_DEV_FBC_SESSIONS_INFO 508

Information about active frame buffer capture sessions on a target device

#define DCGM_FI_DEV_VGPU_VM_ID 520

VM ID of the vGPU instance

#define DCGM_FI_DEV_VGPU_VM_NAME 521

VM name of the vGPU instance

#define DCGM_FI_DEV_VGPU_TYPE 522

vGPU type of the vGPU instance

#define DCGM_FI_DEV_VGPU_UUID 523

UUID of the vGPU instance

#define DCGM_FI_DEV_VGPU_DRIVER_VERSION 524

Driver version of the vGPU instance

#define DCGM_FI_DEV_VGPU_MEMORY_USAGE 525

Memory usage of the vGPU instance

#define DCGM_FI_DEV_VGPU_LICENSE_STATUS 526

License status of the vGPU instance

#define DCGM_FI_DEV_VGPU_FRAME_RATE_LIMIT 527

Frame rate limit of the vGPU instance

#define DCGM_FI_DEV_VGPU_ENC_STATS 528

Current encoder statistics of the vGPU instance

#define DCGM_FI_DEV_VGPU_ENC_SESSIONS_INFO 529

Information about all active encoder sessions on the vGPU instance

#define DCGM_FI_DEV_VGPU_FBC_STATS 530

Statistics of current active frame buffer capture sessions on the vGPU instance

#define DCGM_FI_DEV_VGPU_FBC_SESSIONS_INFO 531

Information about active frame buffer capture sessions on the vGPU instance

```
#define DCGM_FI_FIRST_VGPU_FIELD_ID 520
```

Starting field ID of the vGPU instance

```
#define DCGM_FI_LAST_VGPU_FIELD_ID 570
```

Last field ID of the vGPU instance

```
#define DCGM_FI_MAX_VGPU_FIELDS  
DCGM_FI_LAST_VGPU_FIELD_ID -  
DCGM_FI_FIRST_VGPU_FIELD_ID
```

For now max vGPU field IDs taken as difference of DCGM_FI_LAST_VGPU_FIELD_ID and DCGM_FI_FIRST_VGPU_FIELD_ID i.e. 50

```
#define DCGM_FI_INTERNAL_FIELDS_0_START 600
```

Starting ID for all the internal fields

```
#define DCGM_FI_INTERNAL_FIELDS_0_END 699
```

Last ID for all the internal fields

NVSwitch entity field IDs start here.

NVSwitch latency bins for port 0

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P00  
700
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P00  
701
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P00  
702
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P00  
703
```

Max latency bin

NVSwitch latency bins for port 1

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P01  
704
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P01  
705
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P01  
706
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P01  
707
```

Max latency bin

NVSwitch latency bins for port 2

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P02  
708
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P02  
709
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P02  
710
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P02  
711
```

Max latency bin

NVSwitch latency bins for port 3

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P03  
712
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P03  
713
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P03  
714
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P03  
715
```

Max latency bin

NVSwitch latency bins for port 4

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P04  
716
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P04  
717
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P04  
718
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P04  
719
```

Max latency bin

NVSwitch latency bins for port 5

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P05  
720
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P05  
721
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P05  
722
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P05  
723
```

Max latency bin

NVSwitch latency bins for port 6

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P06  
724
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P06  
725
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P06  
726
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P06  
727
```

Max latency bin

NVSwitch latency bins for port 7

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P07  
728
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P07  
729
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P07  
730
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P07  
731
```

Max latency bin

NVSwitch latency bins for port 8

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P08  
732
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P08  
733
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P08  
734
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P08  
735
```

Max latency bin

NVSwitch latency bins for port 9

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P09  
736
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P09  
737
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P09  
738
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P09  
739
```

Max latency bin

NVSwitch latency bins for port 10

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P10  
740
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P10  
741
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P10  
742
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P10  
743
```

Max latency bin

NVSwitch latency bins for port 11

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P11  
744
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P11  
745
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P11  
746
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P11  
747
```

Max latency bin

NVSwitch latency bins for port 12

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P12  
748
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P12  
749
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P12  
750
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P12  
751
```

Max latency bin

NVSwitch latency bins for port 13

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P13  
752
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P13  
753
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P13  
754
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P13  
755
```

Max latency bin

NVSwitch latency bins for port 14

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P14  
756
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P14  
757
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P14  
758
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P14  
759
```

Max latency bin

NVSwitch latency bins for port 15

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P15  
760
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P15  
761
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P15  
762
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P15  
763
```

Max latency bin

NVSwitch latency bins for port 16

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P16  
764
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P16  
765
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P16  
766
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P16  
767
```

Max latency bin

NVSwitch latency bins for port 17

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_LOW_P17  
768
```

Low latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MED_P17  
769
```

Medium latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_HIGH_P17  
770
```

High latency bin

```
#define DCGM_FI_DEV_NVSWITCH_LATENCY_MAX_P17  
771
```

Max latency bin

NVSwitch Tx and Rx Counter 0 for each port

By default, Counter 0 counts bytes.

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P00 780
```

NVSwitch Tx Bandwidth Counter 0 for port 0

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P00 781
```

NVSwitch Rx Bandwidth Counter 0 for port 0

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P01 782
```

NVSwitch Tx Bandwidth Counter 0 for port 1

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P01 783
```

NVSwitch Rx Bandwidth Counter 0 for port 1

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P02 784
```

NVSwitch Tx Bandwidth Counter 0 for port 2

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P02 785
```

NVSwitch Rx Bandwidth Counter 0 for port 2

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P03 786
```

NVSwitch Tx Bandwidth Counter 0 for port 3

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P03 787
```

NVSwitch Rx Bandwidth Counter 0 for port 3

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P04 788
```

NVSwitch Tx Bandwidth Counter 0 for port 4

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P04 789
```

NVSwitch Rx Bandwidth Counter 0 for port 4

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P05 790
```

NVSwitch Tx Bandwidth Counter 0 for port 5

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P05 791
```

NVSwitch Rx Bandwidth Counter 0 for port 5

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P06 792
```

NVSwitch Tx Bandwidth Counter 0 for port 6

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P06 793
```

NVSwitch Rx Bandwidth Counter 0 for port 6

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P07 794
```

NVSwitch Tx Bandwidth Counter 0 for port 7

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P07 795
```

NVSwitch Rx Bandwidth Counter 0 for port 7

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P08 796
```

NVSwitch Tx Bandwidth Counter 0 for port 8

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P08 797
```

NVSwitch Rx Bandwidth Counter 0 for port 8

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P09 798
```

NVSwitch Tx Bandwidth Counter 0 for port 9

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P09 799
```

NVSwitch Rx Bandwidth Counter 0 for port 9

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P10 800
```

NVSwitch Tx Bandwidth Counter 0 for port 10

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P10 801
```

NVSwitch Rx Bandwidth Counter 0 for port 10

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P11 802
```

NVSwitch Tx Bandwidth Counter 0 for port 11

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P11 803
```

NVSwitch Rx Bandwidth Counter 0 for port 11

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P12 804
```

NVSwitch Tx Bandwidth Counter 0 for port 12

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P12 805
```

NVSwitch Rx Bandwidth Counter 0 for port 12

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P13 806
```

NVSwitch Tx Bandwidth Counter 0 for port 13

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P13 807
```

NVSwitch Rx Bandwidth Counter 0 for port 13

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P14 808
```

NVSwitch Tx Bandwidth Counter 0 for port 14

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P14 809
```

NVSwitch Rx Bandwidth Counter 0 for port 14

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P15 810
```

NVSwitch Tx Bandwidth Counter 0 for port 15

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P15 811
```

NVSwitch Rx Bandwidth Counter 0 for port 15

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P16 812
```

NVSwitch Tx Bandwidth Counter 0 for port 16

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P16 813
```

NVSwitch Rx Bandwidth Counter 0 for port 16

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_0_P17 814
```

NVSwitch Tx Bandwidth Counter 0 for port 17

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_0_P17 815
```

NVSwitch Rx Bandwidth Counter 0 for port 17

NVSwitch Tx and RX Bandwidth Counter 1 for each port

By default, Counter 1 counts packets.

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P00 820
```

NVSwitch Tx Bandwidth Counter 1 for port 0

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P00 821
```

NVSwitch Rx Bandwidth Counter 1 for port 0

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P01 822
```

NVSwitch Tx Bandwidth Counter 1 for port 1

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P01 823
```

NVSwitch Rx Bandwidth Counter 1 for port 1

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P02 824
```

NVSwitch Tx Bandwidth Counter 1 for port 2

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P02 825
```

NVSwitch Rx Bandwidth Counter 1 for port 2

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P03 826
```

NVSwitch Tx Bandwidth Counter 1 for port 3

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P03 827
```

NVSwitch Rx Bandwidth Counter 1 for port 3

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P04 828
```

NVSwitch Tx Bandwidth Counter 1 for port 4

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P04 829
```

NVSwitch Rx Bandwidth Counter 1 for port 4

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P05 830
```

NVSwitch Tx Bandwidth Counter 1 for port 5

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P05 831
```

NVSwitch Rx Bandwidth Counter 1 for port 5

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P06 832
```

NVSwitch Tx Bandwidth Counter 1 for port 6

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P06 833
```

NVSwitch Rx Bandwidth Counter 1 for port 6

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P07 834
```

NVSwitch Tx Bandwidth Counter 1 for port 7

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P07 835
```

NVSwitch Rx Bandwidth Counter 1 for port 7

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P08 836
```

NVSwitch Tx Bandwidth Counter 1 for port 8

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P08 837
```

NVSwitch Rx Bandwidth Counter 1 for port 8

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P09 838
```

NVSwitch Tx Bandwidth Counter 1 for port 9

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P09 839
```

NVSwitch Rx Bandwidth Counter 1 for port 9

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P10 840
```

NVSwitch Tx Bandwidth Counter 0 for port 10

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P10 841
```

NVSwitch Rx Bandwidth Counter 1 for port 10

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P11 842
```

NVSwitch Tx Bandwidth Counter 1 for port 11

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P11 843
```

NVSwitch Rx Bandwidth Counter 1 for port 11

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P12 844
```

NVSwitch Tx Bandwidth Counter 1 for port 12

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P12 845
```

NVSwitch Rx Bandwidth Counter 1 for port 12

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P13 846
```

NVSwitch Tx Bandwidth Counter 0 for port 13

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P13 847
```

NVSwitch Rx Bandwidth Counter 1 for port 13

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P14 848
```

NVSwitch Tx Bandwidth Counter 1 for port 14

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P14 849
```

NVSwitch Rx Bandwidth Counter 1 for port 14

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P15 850
```

NVSwitch Tx Bandwidth Counter 1 for port 15

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P15 851
```

NVSwitch Rx Bandwidth Counter 1 for port 15

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P16 852
```

NVSwitch Tx Bandwidth Counter 1 for port 16

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P16 853
```

NVSwitch Rx Bandwidth Counter 1 for port 16

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_TX_1_P17 854
```

NVSwitch Tx Bandwidth Counter 1 for port 17

```
#define  
DCGM_FI_DEV_NVSWITCH_BANDWIDTH_RX_1_P17 855
```

NVSwitch Rx Bandwidth Counter 1 for port 17

NVSwitch error counters

```
#define DCGM_FI_DEV_NVSWITCH_FATAL_ERRORS 856
```

NVSwitch fatal error information. Note: value field indicates the specific SXid reported

```
#define DCGM_FI_DEV_NVSWITCH_NON_FATAL_ERRORS  
857
```

NVSwitch non fatal error information. Note: value field indicates the specific SXid reported

```
#define DCGM_FI_FIRST_NVSWITCH_FIELD_ID 700
```

Starting field ID of the NVSwitch instance

```
#define DCGM_FI_LAST_NVSWITCH_FIELD_ID 860
```

Last field ID of the NVSwitch instance

```
#define DCGM_FI_MAX_NVSWITCH_FIELDS  
DCGM_FI_LAST_NVSWITCH_FIELD_ID -  
DCGM_FI_FIRST_NVSWITCH_FIELD_ID + 1
```

For now max NVSwitch field Ids taken as difference
of DCGM_FI_LAST_NVSWITCH_FIELD_ID and
DCGM_FI_FIRST_NVSWITCH_FIELD_ID + 1 i.e. 200

```
#define DCGM_FI_MAX_FIELDS 861
```

1 greater than maximum fields above. This is the 1 greater than the maximum field id
that could be allocated

2.19. DCGMAPI_Admin_ExecCtrl

`dcgmReturn_t dcgmUpdateAllFields (dcgmHandle_t pDcgmHandle, int waitForUpdate)`

Parameters

`pDcgmHandle`

IN: DCGM Handle

`waitForUpdate`

IN: Whether or not to wait for the update loop to complete before returning to the
caller 1=wait. 0=do not wait.

Returns

- ▶ DCGM_ST_OK if the call was successful
- ▶ DCGM_ST_BADPARAM if waitForUpdate is invalid
- ▶ DCGM_ST_GENERIC_ERROR if an unspecified DCGM error occurs

Description

This method is used to tell the DCGM module to update all the fields being watched.

Note: If the if the operation mode was set to manual mode (DCGM_OPERATION_MODE_MANUAL) during initialization ([dcgmInit](#)), this method must be caused periodically to allow field value watches the opportunity to gather samples.

dcgmReturn_t dcgmPolicyTrigger (dcgmHandle_t pDcgmHandle)

Parameters

pDcgmHandle

IN: DCGM Handle

Returns

- ▶ DCGM_ST_OK If the call was successful
- ▶ DCGM_ST_GENERIC_ERROR The policy manager was unable to perform another iteration.

Description

Inform the policy manager loop to perform an iteration and trigger the callbacks of any registered functions. Callback functions will be called from a separate thread as the calling function.

Note: The GPU monitoring and management agent must call this method periodically if the operation mode is set to manual mode (DCGM_OPERATION_MODE_MANUAL) during initialization ([dcgmInit](#)).

Chapter 3. DATA STRUCTURES

Here are the data structures with brief descriptions:

`dcgm_field_meta_t`
`dcgm_field_output_format_t`
`dcgmClockSet_v1`
`dcgmConfig_v1`
`dcgmConfigPerfStateSettings_t`
`dcgmConfigPowerLimit_t`
`dcgmConnectV2Params_v1`
`dcgmConnectV2Params_v2`
`dcgmDeviceAttributes_v1`
`dcgmDeviceEncStats_v1`
`dcgmDeviceFbcSessionInfo_v1`
`dcgmDeviceFbcSessions_v1`
`dcgmDeviceFbcStats_v1`
`dcgmDeviceIdentifiers_v1`
`dcgmDeviceMemoryUsage_v1`
`dcgmDevicePidAccountingStats_v1`
`dcgmDevicePowerLimits_v1`
`dcgmDeviceSupportedClockSets_v1`
`dcgmDeviceThermals_v1`
`dcgmDeviceTopology_v1`
`dcgmDeviceVgpuEncSessions_v1`
`dcgmDeviceVgpuIds_v1`
`dcgmDeviceVgpuProcessUtilInfo_v1`
`dcgmDeviceVgpuTypeInfo_v1`
`dcgmDeviceVgpuUtilInfo_v1`
`dcgmDiagResponse_v3`
`dcgmDiagResponsePerGpu_t`
`dcgmErrorInfo_t`
`dcgmFieldGroupInfo_v1`

dcgmFieldValue_v1
dcgmFieldValue_v2
dcgmGpuUsageInfo_t
dcgmGroupEntityPair_t
dcgmGroupInfo_v1
dcgmGroupInfo_v2
dcgmGroupTopology_v1
dcgmHealthResponse_v1
dcgmHealthResponse_v2
dcgmIntrospectContext_v1
dcgmIntrospectCpuUtil_v1
dcgmIntrospectFieldsExecTime_v1
dcgmIntrospectFullFieldsExecTime_v1
dcgmIntrospectFullMemory_v1
dcgmIntrospectMemory_v1
dcgmJobInfo_v2
dcgmModuleGetStatusesModule_t
dcgmNvLinkGpuLinkStatus_t
dcgmNvLinkNvSwitchLinkStatus_t
dcgmNvLinkStatus_v1
dcgmPidInfo_v1
dcgmPidSingleInfo_t
dcgmPolicy_v1
dcgmPolicyCallbackResponse_v1
dcgmPolicyConditionDbe_t
dcgmPolicyConditionMpr_t
dcgmPolicyConditionNvlink_t
dcgmPolicyConditionParms_t
dcgmPolicyConditionPci_t
dcgmPolicyConditionPower_t
dcgmPolicyConditionThermal_t
dcgmPolicyConditionXID_t
dcgmPolicyViolationNotify_t
dcgmProcessUtilInfo_t
dcgmProcessUtilSample_t
dcgmRunningProcess_v1
dcgmStatSummaryFp64_t
dcgmStatSummaryInt32_t
dcgmStatSummaryInt64_t
dcgmVgpuConfig_v1
dcgmVgpuDeviceAttributes_v5
dcgmVgpuInstanceAttributes_v1

3.1. dcgm_field_meta_t Struct Reference

Structure to store meta data for the field

3.2. dcgm_field_output_format_t Struct Reference

Structure for formating the output for dmon. Used as a member in dcgm_field_meta_p

3.3. dcgmClockSet_v1 Struct Reference

Represents a set of memory, SM, and video clocks for a device. This can be current values or a target values based on context

int dcgmClockSet_v1::version

Version Number (dcgmClockSet_version).

unsigned int dcgmClockSet_v1::memClock

Memory Clock (Memory Clock value OR DCGM_INT32_BLANK to Ignore/Use compatible value with smClk).

unsigned int dcgmClockSet_v1::smClock

SM Clock (SM Clock value OR DCGM_INT32_BLANK to Ignore/Use compatible value with memClk).

3.4. dcgmConfig_v1 Struct Reference

Structure to represent default and target configuration for a device

unsigned int dcgmConfig_v1::version

Version number (dcgmConfig_version).

unsigned int dcgmConfig_v1::gpuld

GPU ID.

unsigned int dcgmConfig_v1::eccMode

ECC Mode (0: Disabled, 1 : Enabled, DCGM_INT32_BLANK : Ignored).

unsigned int dcgmConfig_v1::computeMode

Compute Mode (One of DCGM_CONFIG_COMPUTEMODE_? OR DCGM_INT32_BLANK to Ignore).

struct dcgmConfigPerfStateSettings_t**dcgmConfig_v1::perfState**

Performance State Settings (clocks / boost mode).

struct dcgmConfigPowerLimit_t**dcgmConfig_v1::powerLimit**

Power Limits.

3.5. dcgmConfigPerfStateSettings_t Struct Reference

Used to represent Performance state settings

unsigned int dcgmConfigPerfStateSettings_t::syncBoost

Sync Boost Mode (0: Disabled, 1 : Enabled, DCGM_INT32_BLANK : Ignored). Note that using this setting may result in lower clocks than targetClocks.

struct dcgmClockSet_t**dcgmConfigPerfStateSettings_t::targetClocks**

Target clocks. Set smClock and memClock to DCGM_INT32_BLANK to ignore/use compatible values. For GPUs > Maxwell, setting this implies autoBoost=0.

3.6. dcgmConfigPowerLimit_t Struct Reference

Used to represents the power capping limit for each GPU in the group or to represent the power budget for the entire group

dcgmConfigPowerLimitType_t**dcgmConfigPowerLimit_t::type**

Flag to represent power cap for each GPU or power budget for the group of GPUs.

unsigned int dcgmConfigPowerLimit_t::val

Power Limit in Watts (Set a value OR DCGM_INT32_BLANK to Ignore).

3.7. dcgmConnectV2Params_v1 Struct Reference

Connection options for dcgmConnect_v2 (v1)

NOTE: This version is deprecated. use dcgmConnectV2Params_v2

unsigned int dcgmConnectV2Params_v1::version

Version number. Use dcgmConnectV2Params_version.

unsigned int**dcgmConnectV2Params_v1::persistAfterDisconnect****Description**

Whether to persist DCGM state modified by this connection once the connection is terminated. Normally, all field watches created by a connection are removed once a connection goes away. 1 = do not clean up after this connection. 0 = clean up after this connection

3.8. dcgmConnectV2Params_v2 Struct Reference

Connection options for dcgmConnect_v2 (v2)

unsigned int dcgmConnectV2Params_v2::version

Version number. Use dcgmConnectV2Params_version.

unsigned int

dcgmConnectV2Params_v2::persistAfterDisconnect

Description

Whether to persist DCGM state modified by this connection once the connection is terminated. Normally, all field watches created by a connection are removed once a connection goes away. 1 = do not clean up after this connection. 0 = clean up after this connection

unsigned int dcgmConnectV2Params_v2::timeoutMs

Description

When attempting to connect to the specified host engine, how long should we wait in milliseconds before giving up

unsigned int

dcgmConnectV2Params_v2::addressIsUnixSocket

Description

Whether or not the passed-in address is a unix socket filename (1) or a TCP/IP address (0)

3.9. dcgmDeviceAttributes_v1 Struct Reference

Represents attributes corresponding to a device

unsigned int dcgmDeviceAttributes_v1::version

Version number (dcgmDeviceAttributes_version).

struct dcgmDeviceSupportedClockSets_t**dcgmDeviceAttributes_v1::clockSets**

Supported clocks for the device.

struct dcgmDeviceThermals_t**dcgmDeviceAttributes_v1::thermalSettings**

Thermal settings for the device.

struct dcgmDevicePowerLimits_t**dcgmDeviceAttributes_v1::powerLimits**

Various power limits for the device.

struct dcgmDeviceIdentifiers_t**dcgmDeviceAttributes_v1::identifiers**

Identifiers for the device.

struct dcgmDeviceMemoryUsage_t**dcgmDeviceAttributes_v1::memoryUsage**

Memory usage info for the device.

struct dcgmDeviceVgpulds_t**dcgmDeviceAttributes_v1::unusedVgpulds**

Unused Field.

unsigned int**dcgmDeviceAttributes_v1::unusedActiveVgpuInstanceCount**

Unused Field.

unsigned int**dcgmDeviceAttributes_v1::unusedVgpuInstanceIds**

Unused Field.

3.10. dcgmDeviceEncStats_v1 Struct Reference

Represents current encoder statistics for the given device/vGPU instance

unsigned int dcgmDeviceEncStats_v1::version

Version Number (dcgmDeviceEncStats_version).

unsigned int dcgmDeviceEncStats_v1::sessionCount

Count of active encoder sessions.

unsigned int dcgmDeviceEncStats_v1::averageFps

Trailing average FPS of all active sessions.

unsigned int dcgmDeviceEncStats_v1::averageLatency

Encode latency in milliseconds.

3.11. dcgmDeviceFbcSessionInfo_v1 Struct Reference

Represents information about active FBC session on the given device/vGPU instance

unsigned int dcgmDeviceFbcSessionInfo_v1::version

Version Number (dcgmDeviceFbcSessionInfo_version).

unsigned int dcgmDeviceFbcSessionInfo_v1::sessionId

Unique session ID.

unsigned int dcgmDeviceFbcSessionInfo_v1::pid

Owning process ID.

unsigned int dcgmDeviceFbcSessionInfo_v1::vgpuld

vGPU instance ID (only valid on vGPU hosts, otherwise zero)

unsigned int**dcgmDeviceFbcSessionInfo_v1::displayOrdinal**

Display identifier.

dcgmFBCSessionType_t**dcgmDeviceFbcSessionInfo_v1::sessionType**

Type of frame buffer capture session.

unsigned int dcgmDeviceFbcSessionInfo_v1::sessionFlags

Session flags.

unsigned int**dcgmDeviceFbcSessionInfo_v1::hMaxResolution**

Max horizontal resolution supported by the capture session.

unsigned int**dcgmDeviceFbcSessionInfo_v1::vMaxResolution**

Max vertical resolution supported by the capture session.

unsigned int dcgmDeviceFbcSessionInfo_v1::hResolution

Horizontal resolution requested by caller in capture call.

unsigned int dcgmDeviceFbcSessionInfo_v1::vResolution

Vertical resolution requested by caller in capture call.

unsigned int dcgmDeviceFbcSessionInfo_v1::averageFps

Moving average new frames captured per second.

unsigned int**dcgmDeviceFbcSessionInfo_v1::averageLatency**

Moving average new frame capture latency in microseconds.

3.12. dcgmDeviceFbcSessions_v1 Struct Reference

Represents all the active FBC sessions on the given device/vGPU instance

unsigned int dcgmDeviceFbcSessions_v1::version

Version Number (dcgmDeviceFbcSessions_version).

unsigned int dcgmDeviceFbcSessions_v1::sessionCount

Count of active FBC sessions.

struct dcgmDeviceFbcSessionInfo_t**dcgmDeviceFbcSessions_v1::sessionInfo**

Info about the active FBC session.

3.13. dcgmDeviceFbcStats_v1 Struct Reference

Represents current frame buffer capture sessions statistics for the given device/vGPU instance

unsigned int dcgmDeviceFbcStats_v1::version

Version Number (dcgmDeviceFbcStats_version).

unsigned int dcgmDeviceFbcStats_v1::sessionCount

Count of active FBC sessions.

unsigned int dcgmDeviceFbcStats_v1::averageFps

Moving average new frames captured per second.

unsigned int dcgmDeviceFbcStats_v1::averageLatency

Moving average new frame capture latency in microseconds.

3.14. dcgmDeviceIdentifiers_v1 Struct Reference

Represents device identifiers

unsigned int dcgmDeviceIdentifiers_v1::version

Version Number (dcgmDeviceIdentifiers_version).

char dcgmDeviceIdentifiers_v1::brandName

Brand Name.

char dcgmDeviceIdentifiers_v1::deviceName

Name of the device.

char dcgmDeviceIdentifiers_v1::pciBusId

PCI Bus ID.

char dcgmDeviceIdentifiers_v1::serial

Serial for the device.

char dcgmDeviceIdentifiers_v1::uuid

UUID for the device.

char dcgmDeviceIdentifiers_v1::vbios

VBIOS version.

char dcgmDeviceIdentifiers_v1::inforomImageVersion

Inforom Image version.

unsigned int dcgmDeviceIdentifiers_v1::pciDeviceId

The combined 16-bit device id and 16-bit vendor id.

unsigned int dcgmDeviceIdentifiers_v1::pciSubSystemId

The 32-bit Sub System Device ID.

char dcgmDeviceIdentifiers_v1::driverVersion

Driver Version.

unsigned int**dcgmDeviceIdentifiers_v1::virtualizationMode**

Virtualization Mode.

3.15. dcgmDeviceMemoryUsage_v1 Struct Reference

Represents device memory and usage

unsigned int dcgmDeviceMemoryUsage_v1::version

Version Number (dcgmDeviceMemoryUsage_version).

unsigned int dcgmDeviceMemoryUsage_v1::bar1Total

Total BAR1 size in megabytes.

unsigned int dcgmDeviceMemoryUsage_v1::fbTotal

Total framebuffer memory in megabytes.

unsigned int dcgmDeviceMemoryUsage_v1::fbUsed

Used framebuffer memory in megabytes.

unsigned int dcgmDeviceMemoryUsage_v1::fbFree

Free framebuffer memory in megabytes.

3.16. dcgmDevicePidAccountingStats_v1 Struct Reference

Represents accounting data for one process

unsigned int dcgmDevicePidAccountingStats_v1::version

Version Number. Should match dcgmDevicePidAccountingStats_version.

unsigned int dcgmDevicePidAccountingStats_v1::pid

Process id of the process these stats are for.

unsigned int

dcgmDevicePidAccountingStats_v1::gpuUtilization

Description

Percent of time over the process's lifetime during which one or more kernels was executing on the GPU. Set to DCGM_INT32_NOT_SUPPORTED if is not supported

**unsigned int
dcgmDevicePidAccountingStats_v1::memoryUtilization**

Description

Percent of time over the process's lifetime during which global (device) memory was being read or written. Set to DCGM_INT32_NOT_SUPPORTED if is not supported

**unsigned long long
dcgmDevicePidAccountingStats_v1::maxMemoryUsage**

Description

Maximum total memory in bytes that was ever allocated by the process. Set to DCGM_INT64_NOT_SUPPORTED if is not supported

**unsigned long long
dcgmDevicePidAccountingStats_v1::startTimestamp**

CPU Timestamp in usec representing start time for the process.

**unsigned long long
dcgmDevicePidAccountingStats_v1::activeTimeUsec**

Description

Amount of time in usec during which the compute context was active. Note that this does not mean the context was being used. endTimestamp can be computed as startTimestamp + activeTime

3.17. dcgmDevicePowerLimits_v1 Struct Reference

Represents various power limits

unsigned int dcgmDevicePowerLimits_v1::version

Version Number.

unsigned int dcgmDevicePowerLimits_v1::curPowerLimit

Power management limit associated with this device (in W).

unsigned int**dcgmDevicePowerLimits_v1::defaultPowerLimit**

Power management limit effective at device boot (in W).

unsigned int**dcgmDevicePowerLimits_v1::enforcedPowerLimit**

Effective power limit that the driver enforces after taking into account all limiters (in W).

unsigned int**dcgmDevicePowerLimits_v1::minPowerLimit**

Minimum power management limit (in W).

unsigned int**dcgmDevicePowerLimits_v1::maxPowerLimit**

Maximum power management limit (in W).

3.18. dcgmDeviceSupportedClockSets_v1 Struct Reference

Represents list of supported clock sets for a device

unsigned int dcgmDeviceSupportedClockSets_v1::version

Version Number (dcgmDeviceSupportedClockSets_version).

unsigned int dcgmDeviceSupportedClockSets_v1::count

Number of supported clocks.

struct dcgmClockSet_t**dcgmDeviceSupportedClockSets_v1::clockSet**

Valid clock sets for the device. Upto count entries are filled.

3.19. dcgmDeviceThermals_v1 Struct Reference

Represents thermal information

unsigned int dcgmDeviceThermals_v1::version

Version Number.

unsigned int dcgmDeviceThermals_v1::slowdownTemp

Slowdown temperature.

unsigned int dcgmDeviceThermals_v1::shutdownTemp

Shutdown temperature.

3.20. dcgmDeviceTopology_v1 Struct Reference

Device topology information

unsigned int dcgmDeviceTopology_v1::version

version number (dcgmDeviceTopology_version)

unsignedlong dcgmDeviceTopology_v1::cpuAffinityMask**Description**

affinity mask for the specified GPU a 1 represents affinity to the CPU in that bit position
supports up to 256 cores

unsigned int dcgmDeviceTopology_v1::numGpus

number of valid entries in gpuPaths

unsigned int dcgmDeviceTopology_v1::gpuId

gpuId to which the path represents

dcgmGpuTopologyLevel_t dcgmDeviceTopology_v1::path**Description**

path to the gpuId from this GPU. Note that this is a bitmask of DCGM_TOPOLOGY_* values and can contain both PCIe topology and NvLink topology where applicable. For instance: 0x210 = DCGM_TOPOLOGY_CPU | DCGM_TOPOLOGY_NVLINK2 Use the macros DCGM_TOPOLOGY_PATH_NVLINK and DCGM_TOPOLOGY_PATH_PCI to mask the NvLink and PCI paths, respectively.

unsigned int dcgmDeviceTopology_v1::localNvLinkIds**Description**

bits representing the local links connected to gpuId e.g. if this field == 3, links 0 and 1 are connected, field is only valid if NVLINKS actually exist between GPUs

3.21. dcgmDeviceVgpuEncSessions_v1 Struct Reference

Represents information about active encoder sessions on the given vGPU instance

unsigned int dcgmDeviceVgpuEncSessions_v1::version

Version Number (dcgmDeviceVgpuEncSessions_version).

unsigned int dcgmDeviceVgpuEncSessions_v1::vgpuld

vGPU instance ID

unsigned int dcgmDeviceVgpuEncSessions_v1::sessionId

Unique session ID.

unsigned int dcgmDeviceVgpuEncSessions_v1::pid

Process ID.

dcgmEncoderType_t**dcgmDeviceVgpuEncSessions_v1::codecType**

Video encoder type.

unsigned int**dcgmDeviceVgpuEncSessions_v1::hResolution**

Current encode horizontal resolution.

unsigned int**dcgmDeviceVgpuEncSessions_v1::vResolution**

Current encode vertical resolution.

unsigned int**dcgmDeviceVgpuEncSessions_v1::averageFps**

Moving average encode frames per second.

unsigned int**dcgmDeviceVgpuEncSessions_v1::averageLatency**

Moving average encode latency in milliseconds.

3.22. dcgmDeviceVgpuls_v1 Struct Reference

Represents various IDs related to vGPU.

unsigned int dcgmDeviceVgpuids_v1::version

Version Number (dcgmDeviceVgpuIds_version).

unsigned int**dcgmDeviceVgpuids_v1::unusedSupportedVgpuTypeCount**

Unused Field.

unsigned int**dcgmDeviceVgpuids_v1::unusedSupportedVgpuTypeIds**

Unused Field.

unsigned int**dcgmDeviceVgpuids_v1::unusedCreatableVgpuTypeCount**

Unused Field.

unsigned int**dcgmDeviceVgpuids_v1::unusedCreatableVgpuTypeIds**

Unused Field.

3.23. dcgmDeviceVgpuProcessUtilInfo_v1 Struct Reference

Represents utilization values for processes running in vGPU VMs using the device

unsigned int dcgmDeviceVgpuProcessUtilInfo_v1::version

Version Number (dcgmDeviceVgpuProcessUtilInfo_version).

unsigned int dcgmDeviceVgpuProcessUtilInfo_v1::vgpuld

vGPU instance ID

unsigned int**dcgmDeviceVgpuProcessUtilInfo_v1::vgpuProcessSamplesCount**

Count of processes running in the vGPU VM, for which utilization rates are being reported in this cycle.

unsigned int dcgmDeviceVgpuProcessUtilInfo_v1::pid

Process ID of the process running in the vGPU VM.

char dcgmDeviceVgpuProcessUtilInfo_v1::processName

Process Name of process running in the vGPU VM.

unsigned int dcgmDeviceVgpuProcessUtilInfo_v1::smUtil

GPU utilization of process running in the vGPU VM.

unsigned int**dcgmDeviceVgpuProcessUtilInfo_v1::memUtil**

Memory utilization of process running in the vGPU VM.

unsigned int**dcgmDeviceVgpuProcessUtilInfo_v1::encUtil**

Encoder utilization of process running in the vGPU VM.

unsigned int**dcgmDeviceVgpuProcessUtilInfo_v1::decUtil**

Decoder utilization of process running in the vGPU VM.

3.24. dcgmDeviceVgpuTypeInfo_v1 Struct Reference

Represents static info related to vGPUs supported on the device.

unsigned int dcgmDeviceVgpuTypeInfo_v1::version

Version number (dcgmDeviceVgpuTypeIdxStaticInfo_version).

dcgmDeviceVgpuTypeInfo_v1::@2**dcgmDeviceVgpuTypeInfo_v1::vgpuTypeInfo**

vGPU type ID and Supported vGPU type count

char dcgmDeviceVgpuTypeInfo_v1::vgpuTypeName

vGPU type Name

char dcgmDeviceVgpuTypeInfo_v1::vgpuTypeClass

Class of vGPU type.

char dcgmDeviceVgpuTypeInfo_v1::vgpuTypeLicense

license of vGPU type

int dcgmDeviceVgpuTypeInfo_v1::deviceId

device ID of vGPU type

int dcgmDeviceVgpuTypeInfo_v1::subsystemId

Subsystem ID of vGPU type.

int dcgmDeviceVgpuTypeInfo_v1::numDisplayHeads

Count of vGPU's supported display heads.

int dcgmDeviceVgpuTypeInfo_v1::maxInstances

maximum number of vGPU instances creatable on a device for given vGPU type

int dcgmDeviceVgpuTypeInfo_v1::frameRateLimit

Frame rate limit value of the vGPU type.

int dcgmDeviceVgpuTypeInfo_v1::maxResolutionX

vGPU display head's maximum supported resolution in X dimension

int dcgmDeviceVgpuTypeInfo_v1::maxResolutionY

vGPU display head's maximum supported resolution in Y dimension

int dcgmDeviceVgpuTypeInfo_v1::fbTotal

vGPU Total framebuffer size in megabytes

3.25. dcgmDeviceVgpuUtilInfo_v1 Struct Reference

Represents utilization values for vGPUs running on the device

unsigned int dcgmDeviceVgpuUtilInfo_v1::version

Version Number (dcgmDeviceVgpuUtilInfo_version).

unsigned int dcgmDeviceVgpuUtilInfo_v1::vgpuld

vGPU instance ID

unsigned int dcgmDeviceVgpuUtilInfo_v1::smUtil

GPU utilization for vGPU.

unsigned int dcgmDeviceVgpuUtilInfo_v1::memUtil

Memory utilization for vGPU.

unsigned int dcgmDeviceVgpuUtilInfo_v1::encUtil

Encoder utilization for vGPU.

unsigned int dcgmDeviceVgpuUtilInfo_v1::decUtil

Decoder utilization for vGPU.

3.26. dcgmDiagResponse_v3 Struct Reference

Global diagnostics result structure

unsigned int dcgmDiagResponse_v3::version

version number (dcgmDiagResult_version)

unsigned int dcgmDiagResponse_v3::gpuCount

number of valid per GPU results

dcgmDiagResult_t dcgmDiagResponse_v3::blacklist

test for presence of blacklisted drivers (e.g. nouveau)

dcgmDiagResult_t dcgmDiagResponse_v3::nvmlLibrary

test for presence (and version) of NVML lib

dcgmDiagResult_t**dcgmDiagResponse_v3::cudaMainLibrary**

test for presence (and version) of CUDA lib

dcgmDiagResult_t**dcgmDiagResponse_v3::cudaRuntimeLibrary**

test for presence (and version) of CUDA RT lib

dcgmDiagResult_t dcgmDiagResponse_v3::permissions

test for character device permissions

dcgmDiagResult_t**dcgmDiagResponse_v3::persistenceMode**

test for persistence mode enabled

dcgmDiagResult_t dcgmDiagResponse_v3::environment

test for CUDA environment vars that may slow tests

dcgmDiagResult_t**dcgmDiagResponse_v3::pageRetirement**

test for pending frame buffer page retirement

dcgmDiagResult_t dcgmDiagResponse_v3::inforom

test for inforom corruption

dcgmDiagResult_t
dcgmDiagResponse_v3::graphicsProcesses
test for graphics processes running

struct dcgmDiagResponsePerGpu_t
dcgmDiagResponse_v3::perGpuResponses
per GPU test results

char dcgmDiagResponse_v3::systemError
System-wide error reported from NVVS.

3.27. dcgmDiagResponsePerGpu_t Struct Reference

Per GPU diagnostics result structure

unsigned int dcgmDiagResponsePerGpu_t::gpuld
ID for the GPU this information pertains.

unsigned int
dcgmDiagResponsePerGpu_t::hwDiagnosticReturn
Per GPU hardware diagnostic test return code.

dcgmDiagTestResult_t
dcgmDiagResponsePerGpu_t::results
Array with a result for each per-gpu test.

3.28. dcgmErrorInfo_t Struct Reference

Structure to represent error attributes

unsigned int dcgmErrorInfo_t::gpuld

Represents GPU ID.

short dcgmErrorInfo_t::fieldId

One of DCGM_FI_?

int dcgmErrorInfo_t::status

One of DCGM_ST_?

3.29. dcgmFieldGroupInfo_v1 Struct Reference

Structure to represent information about a field group

unsigned int dcgmFieldGroupInfo_v1::version

Version number (dcgmFieldGroupInfo_version).

unsigned int dcgmFieldGroupInfo_v1::numFieldIds

Number of entries in fieldIds[] that are valid.

dcgmFieldGrp_t dcgmFieldGroupInfo_v1::fieldGroupId

ID of this field group.

char dcgmFieldGroupInfo_v1::fieldGroupName

Field Group Name.

unsigned short dcgmFieldGroupInfo_v1::fieldIds

Field ids that belong to this group.

3.30. dcgmFieldValue_v1 Struct Reference

This structure is used to represent value for the field to be queried.

unsigned int dcgmFieldValue_v1::version

version number (dcgmFieldValue_version1)

unsigned short dcgmFieldValue_v1::fieldId

One of DCGM_FI_?

unsigned short dcgmFieldValue_v1::fieldType

One of DCGM_FT_?

int dcgmFieldValue_v1::status

Status for the querying the field. DCGM_ST_OK or one of DCGM_ST_?

int64_t dcgmFieldValue_v1::ts

Timestamp in usec since 1970 */.

int64_t dcgmFieldValue_v1::i64

Int64 value.

double dcgmFieldValue_v1::dbl

Double value.

char dcgmFieldValue_v1::str

NULL terminated string.

char dcgmFieldValue_v1::blob

Binary blob.

dcgmFieldValue_v1::@7 dcgmFieldValue_v1::value

Value.

3.31. dcgmFieldValue_v2 Struct Reference

This structure is used to represent value for the field to be queried.

unsigned int dcgmFieldValue_v2::version

version number (dcgmFieldValue_version2)

dcgm_field_entity_group_t**dcgmFieldValue_v2::entityGroupId**

Entity group this field value's entity belongs to.

dcgm_field_eid_t dcgmFieldValue_v2::entityId

Entity this field value belongs to.

unsigned short dcgmFieldValue_v2::fieldId

One of DCGM_FI_?

unsigned short dcgmFieldValue_v2::fieldType

One of DCGM_FT_?

int dcgmFieldValue_v2::status

Status for the querying the field. DCGM_ST_OK or one of DCGM_ST_?

unsigned int dcgmFieldValue_v2::unused

Unused for now to align ts to an 8-byte boundary. */.

int64_t dcgmFieldValue_v2::ts

Timestamp in usec since 1970 */.

int64_t dcgmFieldValue_v2::i64

Int64 value.

double dcgmFieldValue_v2::dbl

Double value.

char dcgmFieldValue_v2::str

NULL terminated string.

char dcgmFieldValue_v2::blob

Binary blob.

dcgmFieldValue_v2::@8 dcgmFieldValue_v2::value

Value.

3.32. dcgmGpuUsageInfo_t Struct Reference

Info corresponding to the job on a GPU

unsigned int dcgmGpuUsageInfo_t::gpuld

ID of the GPU this pertains to. GPU_ID_INVALID = summary information for multiple GPUs.

long long dcgmGpuUsageInfo_t::energyConsumed

Energy consumed in milliwatt-seconds.

**struct dcgmStatSummaryFp64_t
dcgmGpuUsageInfo_t::powerUsage**

Power usage Min/Max/Avg in watts.

**struct dcgmStatSummaryInt64_t
dcgmGpuUsageInfo_t::pcieRxBandwidth**

PCI-E bytes read from the GPU.

**struct dcgmStatSummaryInt64_t
dcgmGpuUsageInfo_t::pcieTxBandwidth**

PCI-E bytes written to the GPU.

long long dcgmGpuUsageInfo_t::pcieReplays

Count of PCI-E replays that occurred.

long long dcgmGpuUsageInfo_t::startTime

User provided job start time in microseconds since 1970.

long long dcgmGpuUsageInfo_t::endTime

User provided job end time in microseconds since 1970.

**struct dcgmStatSummaryInt32_t
dcgmGpuUsageInfo_t::smUtilization**

GPU SM Utilization in percent.

**struct dcgmStatSummaryInt32_t
dcgmGpuUsageInfo_t::memoryUtilization**

GPU Memory Utilization in percent.

unsigned int dcgmGpuUsageInfo_t::eccSingleBit

Count of ECC single bit errors that occurred.

unsigned int dcgmGpuUsageInfo_t::eccDoubleBit

Count of ECC double bit errors that occurred.

**struct dcgmStatSummaryInt32_t
dcgmGpuUsageInfo_t::memoryClock**

Memory clock in MHz.

**struct dcgmStatSummaryInt32_t
dcgmGpuUsageInfo_t::smClock**

SM clock in MHz.

int dcgmGpuUsageInfo_t::numXidCriticalErrors

Number of valid entries in xidCriticalErrorsTs.

long long dcgmGpuUsageInfo_t::xidCriticalErrorsTs

Timestamps of the critical XID errors that occurred.

int dcgmGpuUsageInfo_t::numComputePids

Count of computePids entries that are valid.

**struct dcgmProcessUtilInfo_t
dcgmGpuUsageInfo_t::computePidInfo**

List of compute processes that ran during the job. 0=no process.

int dcgmGpuUsageInfo_t::numGraphicsPids

Count of graphicsPids entries that are valid.

**struct dcgmProcessUtilInfo_t
dcgmGpuUsageInfo_t::graphicsPidInfo**

List of compute processes that ran during the job. 0=no process.

long long dcgmGpuUsageInfo_t::maxGpuMemoryUsed

Maximum amount of GPU memory that was used in bytes.

long long dcgmGpuUsageInfo_t::powerViolationTime

Number of microseconds we were at reduced clocks due to power violation.

long long dcgmGpuUsageInfo_t::thermalViolationTime

Number of microseconds we were at reduced clocks due to thermal violation.

long long dcgmGpuUsageInfo_t::reliabilityViolationTime

Amount of microseconds we were at reduced clocks due to the reliability limit.

long long**dcgmGpuUsageInfo_t::boardLimitViolationTime**

Amount of microseconds we were at reduced clocks due to being at the board's max voltage.

long long dcgmGpuUsageInfo_t::lowUtilizationTime

Amount of microseconds we were at reduced clocks due to low utilization.

long long dcgmGpuUsageInfo_t::syncBoostTime

Amount of microseconds we were at reduced clocks due to sync boost.

dcgmHealthWatchResults_t**dcgmGpuUsageInfo_t::overallHealth**

The overall health of the system . dcgmHealthWatchResults_t.

dcgmHealthSystems_t dcgmGpuUsageInfo_t::system

system to which this information belongs

dcgmHealthWatchResults_t**dcgmGpuUsageInfo_t::health**

health of the specified system on this GPU

3.33. dcgmGroupEntityPair_t Struct Reference

Represents a entityGroupId + entityId pair to uniquely identify a given entityId inside a group of entities

dcgm_field_entity_group_t
dcgmGroupEntityPair_t::entityGroupId
Entity Group ID entity belongs to.

dcgm_field_eid_t dcgmGroupEntityPair_t::entityId
Entity ID of the entity.

3.34. dcgmGroupInfo_v1 Struct Reference

Structure to store information for DCGM group

unsigned int dcgmGroupInfo_v1::version
Version Number (use dcgmGroupInfo_version1).

unsigned int dcgmGroupInfo_v1::count
count of GPU IDs returned in gpuIdList

unsigned int dcgmGroupInfo_v1::gpuldList
List of GPU Ids part of the group.

char dcgmGroupInfo_v1::groupName
Group Name.

3.35. dcgmGroupInfo_v2 Struct Reference

Structure to store information for DCGM group

unsigned int dcgmGroupInfo_v2::version

Version Number (use dcgmGroupInfo_version2).

unsigned int dcgmGroupInfo_v2::count

count of entityIds returned in entityList

char dcgmGroupInfo_v2::groupName

Group Name.

**struct dcgmGroupEntityPair_t
dcgmGroupInfo_v2::entityList**

List of the entities that are in this group.

3.36. dcgmGroupTopology_v1 Struct Reference

Group topology information

unsigned int dcgmGroupTopology_v1::version

version number (dcgmGroupTopology_version)

unsignedlong**dcgmGroupTopology_v1::groupCpuAffinityMask****Description**

the CPU affinity mask for all GPUs in the group a 1 represents affinity to the CPU in that bit position supports up to 256 cores

unsigned int dcgmGroupTopology_v1::numaOptimalFlag**Description**

a zero value indicates that 1 or more GPUs in the group have a different CPU affinity and thus may not be optimal for certain algorithms

dcgmGpuTopologyLevel_t
dcgmGroupTopology_v1::slowestPath
the slowest path amongst GPUs in the group

3.37. dcgmHealthResponse_v1 Struct Reference

Health Response structure version 1. GPU Only

unsigned int dcgmHealthResponse_v1::version
version number (dcgmHealthResponse_version)

dcgmHealthWatchResults_t
dcgmHealthResponse_v1::overallHealth

The overall health of the system. dcgmHealthWatchResults_t.

Description

overall health of this GPU

unsigned int dcgmHealthResponse_v1::gpuCount

The number of GPUs with warnings/errors.

unsigned int dcgmHealthResponse_v1::gpuld

GPU ID for which this data is valid.

unsigned int dcgmHealthResponse_v1::incidentCount

The number of systems that encountered a warning/error.

dcgmHealthSystems_t dcgmHealthResponse_v1::system

system to which this information belongs

dcgmHealthWatchResults_t**dcgmHealthResponse_v1::health**

health of the specified system on this GPU

char dcgmHealthResponse_v1::errorString

information about the error(s) or warning(s) flagged

3.38. dcgmHealthResponse_v2 Struct Reference

Health Response structure version 2 - NvSwitch-compatible

unsigned int dcgmHealthResponse_v2::version

version number (dcgmHealthResponse_version)

dcgmHealthWatchResults_t**dcgmHealthResponse_v2::overallHealth**

The overall health of the system. dcgmHealthWatchResults_t.

Description

overall health of this entity

unsigned int dcgmHealthResponse_v2::entityCount

The number of entities with warnings/errors.

dcgm_field_entity_group_t**dcgmHealthResponse_v2::entityGroupId**

entity group entityId belongs to

dcgm_field_eid_t dcgmHealthResponse_v2::entityId

entity for which this data is valid

unsigned int dcgmHealthResponse_v2::incidentCount

The number of systems that encountered a warning/error.

dcgmHealthSystems_t dcgmHealthResponse_v2::system

system to which this information belongs

dcgmHealthWatchResults_t**dcgmHealthResponse_v2::health**

health of the specified system on this entity

char dcgmHealthResponse_v2::errorString

information about the error(s) or warning(s) flagged

3.39. dcgmlIntrospectContext_v1 Struct Reference

Identifies the retrieval context for introspection API calls.

unsigned int dcgmlIntrospectContext_v1::version

version number (dcgmIntrospectContext_version)

dcgmlIntrospectLevel_t**dcgmlIntrospectContext_v1::introspectLvl**

Introspect Level dcgmlIntrospectLevel_t.

dcgmGpuGrp_t dcgmlIntrospectContext_v1::fieldGroupId

Only needed if introspectLvl is DCGM_INTROSPECT_LVL_FIELD_GROUP.

unsigned short dcgmlIntrospectContext_v1::fieldId

Only needed if introspectLvl is DCGM_INTROSPECT_LVL_FIELD.

unsigned long long**dcgmlIntrospectContext_v1::contextId**

Overloaded way to access both fieldGroupId and fieldId.

3.40. dcgmlIntrospectCpuUtil_v1 Struct Reference

DCGM CPU Utilization information. Multiply values by 100 to get them in %.

unsigned int dcgmlIntrospectCpuUtil_v1::version

version number (dcgmMetadataCpuUtil_version)

double dcgmlIntrospectCpuUtil_v1::total

fraction of device's CPU resources that were used

double dcgmlIntrospectCpuUtil_v1::kernel

fraction of device's CPU resources that were used in kernel mode

double dcgmlIntrospectCpuUtil_v1::user

fraction of device's CPU resources that were used in user mode

3.41. dcgmlIntrospectFieldsExecTime_v1 Struct Reference

DCGM Execution time info for a set of fields

unsigned int dcgmlIntrospectFieldsExecTime_v1::version

version number (dcgmlIntrospectFieldsExecTime_version)

long long

dcgmlIntrospectFieldsExecTime_v1::meanUpdateFreqUsec

the mean update frequency of all fields

double

dcgmlIntrospectFieldsExecTime_v1::recentUpdateUsec

Description

the sum of every field's most recent execution time after they have been normalized to [meanUpdateFreqUsec](#)". This is roughly how long it takes to update fields every meanUpdateFreqUsec

long long

dcgmlIntrospectFieldsExecTime_v1::totalEverUpdateUsec

The total amount of time, ever, that has been spent updating all the fields.

3.42. dcgmlIntrospectFullFieldsExecTime_v1 Struct Reference

Full introspection info for field execution time

unsigned int

dcgmlIntrospectFullFieldsExecTime_v1::version

version number (dcgmlIntrospectFullFieldsExecTime_version)

struct dcgmlIntrospectFieldsExecTime_v1

dcgmlIntrospectFullFieldsExecTime_v1::aggregateInfo

info that includes global and device scope

int dcgmlIntrospectFullFieldsExecTime_v1::hasGlobalInfo

0 means globalInfo is populated, !0 means it's not

struct dcgmlIntrospectFieldsExecTime_v1

dcgmlIntrospectFullFieldsExecTime_v1::globalInfo

info that only includes global field scope

unsigned short

dcgmlIntrospectFullFieldsExecTime_v1::gpuInfoCount

count of how many entries in gpuInfo are populated

unsigned int

dcgmlIntrospectFullFieldsExecTime_v1::gpuidsForGpuInfo

Description

the GPU ID at a given index identifies which gpu the corresponding entry in **gpuInfo** is from

struct dcgmlIntrospectFieldsExecTime_v1

dcgmlIntrospectFullFieldsExecTime_v1::gpuInfo

Description

info that is separated by the GPU ID that the watches were for

3.43. **dcgmlIntrospectFullMemory_v1** Struct Reference

Full introspection info for field memory

unsigned int dcgmlIntrospectFullMemory_v1::version

version number (dcgmlIntrospectFullMemory_version)

struct dcgmlIntrospectMemory_v1

dcgmlIntrospectFullMemory_v1::aggregateInfo

info that includes global and device scope

int dcgmlIntrospectFullMemory_v1::hasGlobalInfo

0 means globalInfo is populated, !0 means it's not

struct dcgmlIntrospectMemory_v1

dcgmlIntrospectFullMemory_v1::globalInfo

info that only includes global field scope

unsigned short

dcgmlIntrospectFullMemory_v1::gpuInfoCount

count of how many entries in gpuInfo are populated

unsigned int

dcgmlIntrospectFullMemory_v1::gpuidsForGpuInfo

Description

the GPU ID at a given index identifies which gpu the corresponding entry in **gpuInfo** is from

struct dcgmlIntrospectMemory_v1

dcgmlIntrospectFullMemory_v1::gpuInfo

Description

info that is divided by the GPU ID that the watches were for

3.44. dcgmlIntrospectMemory_v1 Struct Reference

DCGM Memory usage information

unsigned int dcgmIntrospectMemory_v1::version

version number (dcgmIntrospectMemory_version)

long long dcgmIntrospectMemory_v1::bytesUsed

number of bytes

3.45. dcgmJobInfo_v2 Struct Reference

To store job statistics The following fields are not applicable in the summary info:

- ▶ pcieRxBandwidth (Min/Max)
- ▶ pcieTxBandwidth (Min/Max)
- ▶ smUtilization (Min/Max)
- ▶ memoryUtilization (Min/Max)
- ▶ memoryClock (Min/Max)
- ▶ smClock (Min/Max)
- ▶ processSamples

The average value in the above fields (in the summary) is the average of the averages of respective fields from all GPUs

unsigned int dcgmJobInfo_v2::version

Version of this message (dcgmPidInfo_version).

int dcgmJobInfo_v2::numGpus

Number of GPUs that are valid in gpus[].

struct dcgmGpuUsageInfo_t dcgmJobInfo_v2::summary

Summary information for all GPUs listed in gpus[].

struct dcgmGpuUsageInfo_t dcgmJobInfo_v2::gpus

Per-GPU information for this PID.

3.46. dcgmModuleGetStatusesModule_t Struct Reference

Status of all of the modules of the host engine

dcgmModuleId_t dcgmModuleGetStatusesModule_t::id

ID of this module.

dcgmModuleStatus_t**dcgmModuleGetStatusesModule_t::status**

Status of this module.

3.47. dcgmNvLinkGpuLinkStatus_t Struct Reference

State of NvLink links for a GPU

dcgm_field_eid_t dcgmNvLinkGpuLinkStatus_t::entityId

Entity ID of the GPU (gpuId).

dcgmNvLinkLinkState_t**dcgmNvLinkGpuLinkStatus_t::linkState**

Per-GPU link states.

3.48. dcgmNvLinkNvSwitchLinkStatus_t Struct Reference

State of NvLink links for a NvSwitch

dcgm_field_eid_t**dcgmNvLinkNvSwitchLinkStatus_t::entityId**

Entity ID of the NvSwitch (physicalId).

dcgmNvLinkLinkState_t**dcgmNvLinkNvSwitchLinkStatus_t::linkState**

Per-NvSwitch link states.

3.49. dcgmNvLinkStatus_v1 Struct Reference

Status of all of the NvLinks in a given system

unsigned int dcgmNvLinkStatus_v1::version

Version of this request. Should be dcgmNvLinkStatus_version1.

unsigned int dcgmNvLinkStatus_v1::numGpus

Number of entries in gpus[] that are populated.

**struct dcgmNvLinkGpuLinkStatus_t
dcgmNvLinkStatus_v1::gpus**

Per-GPU NvLink link statuses.

unsigned int dcgmNvLinkStatus_v1::numNvSwitches

Number of entries in nvSwitches[] that are populated.

**struct dcgmNvLinkNvSwitchLinkStatus_t
dcgmNvLinkStatus_v1::nvSwitches**

Per-NvSwitch link statuses.

3.50. dcgmPidInfo_v1 Struct Reference

To store process statistics

unsigned int dcgmPidInfo_v1::version

Version of this message (dcgmPidInfo_version).

unsigned int dcgmPidInfo_v1::pid

PID of the process.

int dcgmPidInfo_v1::numGpus

Number of GPUs that are valid in GPUs.

struct dcgmPidSingleInfo_t dcgmPidInfo_v1::summary

Summary information for all GPUs listed in gpus[].

struct dcgmPidSingleInfo_t dcgmPidInfo_v1::gpus

Per-GPU information for this PID.

3.51. dcgmPidSingleInfo_t Struct Reference

Info corresponding to single PID

unsigned int dcgmPidSingleInfo_t::gpuld

ID of the GPU this pertains to. GPU_ID_INVALID = summary information for multiple GPUs.

long long dcgmPidSingleInfo_t::energyConsumed

Energy consumed by the gpu in milliwatt-seconds.

struct dcgmStatSummaryInt64_t**dcgmPidSingleInfo_t::pcieRxBandwidth**

PCI-E bytes read from the GPU.

struct dcgmStatSummaryInt64_t**dcgmPidSingleInfo_t::pcieTxBandwidth**

PCI-E bytes written to the GPU.

long long dcgmPidSingleInfo_t::pcieReplays

Count of PCI-E replays that occurred.

long long dcgmPidSingleInfo_t::startTime

Process start time in microseconds since 1970.

long long dcgmPidSingleInfo_t::endTime

Process end time in microseconds since 1970 or reported as 0 if the process is not completed.

struct dcgmProcessUtilInfo_t**dcgmPidSingleInfo_t::processUtilization**

Process SM and Memory Utilization (in percent).

struct dcgmStatSummaryInt32_t**dcgmPidSingleInfo_t::smUtilization**

GPU SM Utilization in percent.

struct dcgmStatSummaryInt32_t**dcgmPidSingleInfo_t::memoryUtilization**

GPU Memory Utilization in percent.

unsigned int dcgmPidSingleInfo_t::eccSingleBit

Count of ECC single bit errors that occurred.

unsigned int dcgmPidSingleInfo_t::eccDoubleBit

Count of ECC double bit errors that occurred.

**struct dcgmStatSummaryInt32_t
dcgmPidSingleInfo_t::memoryClock**

Memory clock in MHz.

**struct dcgmStatSummaryInt32_t
dcgmPidSingleInfo_t::smClock**

SM clock in MHz.

int dcgmPidSingleInfo_t::numXidCriticalErrors

Number of valid entries in xidCriticalErrorsTs.

long long dcgmPidSingleInfo_t::xidCriticalErrorsTs

Timestamps of the critical XID errors that occurred.

int dcgmPidSingleInfo_t::numOtherComputePids

Count of otherComputePids entries that are valid.

unsigned int dcgmPidSingleInfo_t::otherComputePids

Other compute processes that ran. 0=no process.

int dcgmPidSingleInfo_t::numOtherGraphicsPids

Count of otherGraphicsPids entries that are valid.

unsigned int dcgmPidSingleInfo_t::otherGraphicsPids

Other graphics processes that ran. 0=no process.

long long dcgmPidSingleInfo_t::maxGpuMemoryUsed

Maximum amount of GPU memory that was used in bytes.

long long dcgmPidSingleInfo_t::powerViolationTime

Number of microseconds we were at reduced clocks due to power violation.

long long dcgmPidSingleInfo_t::thermalViolationTime

Number of microseconds we were at reduced clocks due to thermal violation.

long long dcgmPidSingleInfo_t::reliabilityViolationTime

Amount of microseconds we were at reduced clocks due to the reliability limit.

long long dcgmPidSingleInfo_t::boardLimitViolationTime

Amount of microseconds we were at reduced clocks due to being at the board's max voltage.

long long dcgmPidSingleInfo_t::lowUtilizationTime

Amount of microseconds we were at reduced clocks due to low utilization.

long long dcgmPidSingleInfo_t::syncBoostTime

Amount of microseconds we were at reduced clocks due to sync boost.

dcgmHealthWatchResults_t**dcgmPidSingleInfo_t::overallHealth**

The overall health of the system . dcgmHealthWatchResults_t.

dcgmHealthSystems_t dcgmPidSingleInfo_t::system

system to which this information belongs

dcgmHealthWatchResults_t dcgmPidSingleInfo_t::health

health of the specified system on this GPU

3.52. dcgmPolicy_v1 Struct Reference

Define the structure that specifies a policy to be enforced for a GPU

unsigned int dcgmPolicy_v1::version

version number (dcgmPolicy_version)

dcgmPolicyCondition_t dcgmPolicy_v1::condition

Condition(s) to access dcgmPolicyCondition_t.

dcgmPolicyMode_t dcgmPolicy_v1::mode

Mode of operation dcgmPolicyMode_t.

dcgmPolicyIsolation_t dcgmPolicy_v1::isolation

Isolation level after a policy violation dcgmPolicyIsolation_t.

dcgmPolicyAction_t dcgmPolicy_v1::action

Action to perform after a policy violation dcgmPolicyAction_t action.

dcgmPolicyValidation_t dcgmPolicy_v1::validation

Validation to perform after action is taken dcgmPolicyValidation_t.

dcgmPolicyFailureResp_t dcgmPolicy_v1::response

Failure to validation response dcgmPolicyFailureResp_t.

struct dcgmPolicyConditionParms_t**dcgmPolicy_v1::parms**

Parameters for the condition fields.

3.53. dcgmPolicyCallbackResponse_v1 Struct Reference

Define the structure that is given to the callback function

unsigned int dcgmPolicyCallbackResponse_v1::version

version number (dcgmPolicyCallbackResponse_version)

dcgmPolicyCondition_t

dcgmPolicyCallbackResponse_v1::condition

Condition that was violated.

struct dcgmPolicyConditionDbe_t

dcgmPolicyCallbackResponse_v1::dbe

ECC DBE return structure.

struct dcgmPolicyConditionPci_t

dcgmPolicyCallbackResponse_v1::pci

PCI replay error return structure.

struct dcgmPolicyConditionMpr_t

dcgmPolicyCallbackResponse_v1::mpr

Max retired pages limit return structure.

struct dcgmPolicyConditionThermal_t

dcgmPolicyCallbackResponse_v1::thermal

Thermal policy violations return structure.

struct dcgmPolicyConditionPower_t

dcgmPolicyCallbackResponse_v1::power

Power policy violations return structure.

struct dcgmPolicyConditionNvlink_t

dcgmPolicyCallbackResponse_v1::nvlink

Nvlink policy violations return structure.

struct dcgmPolicyConditionXID_t

dcgmPolicyCallbackResponse_v1::xid

XID policy violations return structure.

3.54. dcgmPolicyConditionDbe_t Struct Reference

Define the ECC DBE return structure

long long dcgmPolicyConditionDbe_t::timestamp

timestamp of the error

enum dcgmPolicyConditionDbe_t::@5

dcgmPolicyConditionDbe_t::location

location of the error

unsigned int dcgmPolicyConditionDbe_t::numerrors

number of errors

3.55. dcgmPolicyConditionMpr_t Struct Reference

Define the maximum pending retired pages limit return structure

long long dcgmPolicyConditionMpr_t::timestamp

timestamp of the error

unsigned int dcgmPolicyConditionMpr_t::sbepages

number of pending pages due to SBE

unsigned int dcgmPolicyConditionMpr_t::dbepages

number of pending pages due to DBE

3.56. dcgmPolicyConditionNvlink_t Struct Reference

Define the nvlink policy violations return structure

long long dcgmPolicyConditionNvlink_t::timestamp

timestamp of the error

unsigned short dcgmPolicyConditionNvlink_t::fieldId

Nvlink counter field ID that violated policy.

unsigned int dcgmPolicyConditionNvlink_t::counter

Nvlink counter value that violated policy.

3.57. dcgmPolicyConditionParms_t Struct Reference

Structure for policy condition parameters. This structure contains a tag that represents the type of the value being passed as well as a "val" which is a union of the possible value types. For example, to pass a true boolean: tag = BOOL, val.boolean = 1.

3.58. dcgmPolicyConditionPci_t Struct Reference

Define the PCI replay error return structure

long long dcgmPolicyConditionPci_t::timestamp

timestamp of the error

unsigned int dcgmPolicyConditionPci_t::counter

value of the PCIe replay counter

3.59. dcgmPolicyConditionPower_t Struct Reference

Define the power policy violations return structure

long long dcgmPolicyConditionPower_t::timestamp

timestamp of the error

unsigned int

dcgmPolicyConditionPower_t::powerViolation

Power value reached that violated policy.

3.60. dcgmPolicyConditionThermal_t Struct Reference

Define the thermal policy violations return structure

long long dcgmPolicyConditionThermal_t::timestamp

timestamp of the error

unsigned int

dcgmPolicyConditionThermal_t::thermalViolation

Temperature reached that violated policy.

3.61. dcgmPolicyConditionXID_t Struct Reference

Define the xid policy violations return structure

long long dcgmPolicyConditionXID_t::timestamp

Timestamp of the error.

unsigned int dcgmPolicyConditionXID_t::errnum

The XID error number.

3.62. dcgmPolicyViolationNotify_t Struct Reference

Structure to fill when a user queries for policy violations

unsigned int dcgmPolicyViolationNotify_t::gpuld
gpu ID

unsigned int
dcgmPolicyViolationNotify_t::violationOccurred
a violation based on the bit values in dcgmPolicyCondition_t

3.63. dcgmProcessUtilInfo_t Struct Reference

per process utilization rates

3.64. dcgmProcessUtilSample_t Struct Reference

Internal structure used to get the PID and the corresponding utilization rate

3.65. dcgmRunningProcess_v1 Struct Reference

Running process information for a compute or graphics process

unsigned int dcgmRunningProcess_v1::version

Version of this message (dcgmRunningProcess_version).

unsigned int dcgmRunningProcess_v1::pid

PID of the process.

unsigned long long
dcgmRunningProcess_v1::memoryUsed

GPU memory used by this process in bytes.

3.66. dcgmStatSummaryFp64_t Struct Reference

Summary of time series data in double-precision format. Each value will either be set or be a BLANK value. Check for blank with the `DCGM_FP64_IS_BLANK()` macro. See `dcmvalue.h` for the actual values of BLANK values

double dcgmStatSummaryFp64_t::minValue

Minimum value of the samples looked at.

double dcgmStatSummaryFp64_t::maxValue

Maximum value of the samples looked at.

double dcgmStatSummaryFp64_t::average

Simple average of the samples looked at. Blank values are ignored for this calculation.

3.67. dcgmStatSummaryInt32_t Struct Reference

Same as [dcgmStatSummaryInt64_t](#), but with 32-bit integer values

int dcgmStatSummaryInt32_t::minValue

Minimum value of the samples looked at.

int dcgmStatSummaryInt32_t::maxValue

Maximum value of the samples looked at.

int dcgmStatSummaryInt32_t::average

Simple average of the samples looked at. Blank values are ignored for this calculation.

3.68. dcgmStatSummaryInt64_t Struct Reference

Summary of time series data in int64 format. Each value will either be set or be a BLANK value. Check for blank with the [DCGM_INT64_IS_BLANK\(\)](#) macro. See [dcgmvalue.h](#) for the actual values of BLANK values

long long dcgmStatSummaryInt64_t::minValue

Minimum value of the samples looked at.

long long dcgmStatSummaryInt64_t::maxValue

Maximum value of the samples looked at.

long long dcgmStatSummaryInt64_t::average

Simple average of the samples looked at. Blank values are ignored for this calculation.

3.69. dcgmVgpuConfig_v1 Struct Reference

Structure to represent default and target vgpu configuration for a device

unsigned int dcgmVgpuConfig_v1::version

Version number (dcgmConfig_version).

unsigned int dcgmVgpuConfig_v1::gpuld

GPU ID.

unsigned int dcgmVgpuConfig_v1::eccMode

ECC Mode (0: Disabled, 1 : Enabled, DCGM_INT32_BLANK : Ignored).

unsigned int dcgmVgpuConfig_v1::computeMode

Compute Mode (One of DCGM_CONFIG_COMPUTEMODE_? OR DCGM_INT32_BLANK to Ignore).

struct dcgmConfigPerfStateSettings_t**dcgmVgpuConfig_v1::perfState**

Performance State Settings (clocks / boost mode).

struct dcgmConfigPowerLimit_t**dcgmVgpuConfig_v1::powerLimit**

Power Limits.

3.70. dcgmVgpuDeviceAttributes_v5 Struct Reference

Represents the vGPU attributes corresponding to a physical device

unsigned int dcgmVgpuDeviceAttributes_v5::version

Version number (dcgmVgpuDeviceAttributes_version).

unsigned int**dcgmVgpuDeviceAttributes_v5::activeVgpuInstanceCount**

Count of active vGPU instances on the device.

unsigned int**dcgmVgpuDeviceAttributes_v5::activeVgpuInstancelds**

List of vGPU instances.

unsigned int**dcgmVgpuDeviceAttributes_v5::creatableVgpuTypeCount**

Creatable vGPU type count.

unsigned int**dcgmVgpuDeviceAttributes_v5::creatableVgpuTypelds**

List of Creatable vGPU types.

unsigned int**dcgmVgpuDeviceAttributes_v5::supportedVgpuTypeCount**

Supported vGPU type count.

struct dcgmDeviceVgpuTypeInfo_t**dcgmVgpuDeviceAttributes_v5::supportedVgpuTypeInfo**

Info related to vGPUs supported on the device.

struct dcgmDeviceVgpuUtilInfo_t**dcgmVgpuDeviceAttributes_v5::vgpuUtilInfo**

Utilizations specific to vGPU instance.

unsigned int dcgmVgpuDeviceAttributes_v5::gpuUtil

GPU utilization.

unsigned int**dcgmVgpuDeviceAttributes_v5::memCopyUtil**

Memory utilization.

unsigned int dcgmVgpuDeviceAttributes_v5::encUtil

Encoder utilization.

unsigned int dcgmVgpuDeviceAttributes_v5::decUtil

Decoder utilization.

3.71. dcgmVgpuInstanceAttributes_v1 Struct Reference

Represents attributes specific to vGPU instance

unsigned int dcgmVgpuInstanceAttributes_v1::version

Version number (dcgmVgpuInstanceAttributes_version).

char dcgmVgpuInstanceAttributes_v1::vmlId

VM ID of the vGPU instance.

char dcgmVgpuInstanceAttributes_v1::vmName

VM name of the vGPU instance.

unsigned int**dcgmVgpuInstanceAttributes_v1::vgpuTypeid**

Type ID of the vGPU instance.

char dcgmVgpuInstanceAttributes_v1::vgpuUuid

UUID of the vGPU instance.

char dcgmVgpuInstanceAttributes_v1::vgpuDriverVersion

Driver version of the vGPU instance.

unsigned int dcgmVgpuInstanceAttributes_v1::fbUsage

Fb usage of the vGPU instance.

unsigned int**dcgmVgpuInstanceAttributes_v1::licenseStatus**

License status of the vGPU instance.

unsigned int**dcgmVgpuInstanceAttributes_v1::frameRateLimit**

Frame rate limit of the vGPU instance.

Chapter 4. DATA FIELDS

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

A

action

[dcgmPolicy_v1](#)

activeTimeUsec

[dcgmDevicePidAccountingStats_v1](#)

activeVgpuInstanceCount

[dcgmVgpuDeviceAttributes_v5](#)

activeVgpuInstanceIds

[dcgmVgpuDeviceAttributes_v5](#)

addressIsUnixSocket

[dcgmConnectV2Params_v2](#)

aggregateInfo

[dcgmIntrospectFullFieldsExecTime_v1](#)

[dcgmIntrospectFullMemory_v1](#)

average

[dcgmStatSummaryInt64_t](#)

[dcgmStatSummaryInt32_t](#)

[dcgmStatSummaryFp64_t](#)

averageFps

[dcgmDeviceEncStats_v1](#)

[dcgmDeviceFbcSessionInfo_v1](#)

[dcgmDeviceVgpuEncSessions_v1](#)

[dcgmDeviceFbcStats_v1](#)

averageLatency

[dcgmDeviceFbcStats_v1](#)

[dcgmDeviceFbcSessionInfo_v1](#)

[dcgmDeviceVgpuEncSessions_v1](#)

`dcmDeviceEncStats_v1`

B

bar1Total

`dcmDeviceMemoryUsage_v1`

blacklist

`dcmDiagResponse_v3`

blob

`dcmFieldValue_v2`

`dcmFieldValue_v1`

boardLimitViolationTime

`dcmPidSingleInfo_t`

`dcmGpuUsageInfo_t`

brandName

`dcmDeviceIdentifiers_v1`

bytesUsed

`dcmIntrospectMemory_v1`

C

clockSet

`dcmDeviceSupportedClockSets_v1`

clockSets

`dcmDeviceAttributes_v1`

codecType

`dcmDeviceVgpuEncSessions_v1`

computeMode

`dcmConfig_v1`

`dcmVgpuConfig_v1`

computePidInfo

`dcmGpuUsageInfo_t`

condition

`dcmPolicyCallbackResponse_v1`

`dcmPolicy_v1`

contextId

`dcmIntrospectContext_v1`

count

`dcmGroupInfo_v1`

`dcmGroupInfo_v2`

`dcmDeviceSupportedClockSets_v1`

counter

`dcmPolicyConditionPci_t`

`dcmPolicyConditionNvlink_t`

cpuAffinityMask
 dcgmDeviceTopology_v1
creatableVgpuTypeCount
 dcgmVgpuDeviceAttributes_v5
creatableVgpuTypeIds
 dcgmVgpuDeviceAttributes_v5
cudaMainLibrary
 dcgmDiagResponse_v3
cudaRuntimeLibrary
 dcgmDiagResponse_v3
curPowerLimit
 dcgmDevicePowerLimits_v1

D

dbe
 dcgmPolicyCallbackResponse_v1
dbepages
 dcgmPolicyConditionMpr_t
dbl
 dcgmFieldValue_v2
 dcgmFieldValue_v1
decUtil
 dcgmDeviceVgpuUtilInfo_v1
 dcgmDeviceVgpuProcessUtilInfo_v1
 dcgmVgpuDeviceAttributes_v5
defaultPowerLimit
 dcgmDevicePowerLimits_v1
deviceId
 dcgmDeviceVgpuTypeInfo_v1
deviceName
 dcgmDeviceIdentifiers_v1
displayOrdinal
 dcgmDeviceFbcSessionInfo_v1
driverVersion
 dcgmDeviceIdentifiers_v1

E

eccDoubleBit
 dcgmPidSingleInfo_t
 dcgmGpuUsageInfo_t
eccMode
 dcgmVgpuConfig_v1
 dcgmConfig_v1

eccSingleBit
 dcgmPidSingleInfo_t
 dcgmGpuUsageInfo_t

encUtil
 dcgmDeviceVgpuProcessUtilInfo_v1
 dcgmVgpuDeviceAttributes_v5
 dcgmDeviceVgpuUtilInfo_v1

endTime
 dcgmPidSingleInfo_t
 dcgmGpuUsageInfo_t

energyConsumed
 dcgmPidSingleInfo_t
 dcgmGpuUsageInfo_t

enforcedPowerLimit
 dcgmDevicePowerLimits_v1

entityCount
 dcgmHealthResponse_v2

entityGroupId
 dcgmGroupEntityPair_t
 dcgmFieldValue_v2
 dcgmHealthResponse_v2

entityId
 dcgmNvLinkGpuLinkStatus_t
 dcgmGroupEntityPair_t
 dcgmFieldValue_v2
 dcgmHealthResponse_v2
 dcgmNvLinkNvSwitchLinkStatus_t

entityList
 dcgmGroupInfo_v2

environment
 dcgmDiagResponse_v3

errnum
 dcgmPolicyConditionXID_t

errorString
 dcgmHealthResponse_v2
 dcgmHealthResponse_v1

F

fbFree
 dcgmDeviceMemoryUsage_v1

fbTotal
 dcgmDeviceMemoryUsage_v1
 dcgmDeviceVgpuTypeInfo_v1

fbUsage
`dcmgVgpuInstanceAttributes_v1`

fbUsed
`dcmgDeviceMemoryUsage_v1`

fieldGroupId
`dcmgFieldGroupInfo_v1`
`dcmgIntrospectContext_v1`

fieldGroupName
`dcmgFieldGroupInfo_v1`

fieldId
`dcmgFieldValue_v1`
`dcmgErrorInfo_t`
`dcmgPolicyConditionNvlink_t`
`dcmgIntrospectContext_v1`
`dcmgFieldValue_v2`

fieldIds
`dcmgFieldGroupInfo_v1`

fieldType
`dcmgFieldValue_v1`
`dcmgFieldValue_v2`

frameRateLimit
`dcmgDeviceVgpuTypeInfo_v1`
`dcmgVgpuInstanceAttributes_v1`

G

globalInfo
`dcmgIntrospectFullFieldsExecTime_v1`
`dcmgIntrospectFullMemory_v1`

gpuCount
`dcmgDiagResponse_v3`
`dcmgHealthResponse_v1`

gpuId
`dcmgPolicyViolationNotify_t`
`dcmgHealthResponse_v1`
`dcmgErrorInfo_t`
`dcmgPidSingleInfo_t`
`dcmgGpuUsageInfo_t`
`dcmgConfig_v1`
`dcmgDiagResponsePerGpu_t`
`dcmgDeviceTopology_v1`
`dcmgVgpuConfig_v1`

gpuIdList
`dcmgGroupInfo_v1`

```

gpuIdsForGpuInfo
    dcgmIntrospectFullMemory_v1
    dcgmIntrospectFullFieldsExecTime_v1
gpuInfo
    dcgmIntrospectFullFieldsExecTime_v1
    dcgmIntrospectFullMemory_v1
gpuInfoCount
    dcgmIntrospectFullMemory_v1
    dcgmIntrospectFullFieldsExecTime_v1
gpus
    dcgmJobInfo_v2
    dcgmNvLinkStatus_v1
    dcgmPidInfo_v1
gpuUtil
    dcgmVgpuDeviceAttributes_v5
gpuUtilization
    dcgmDevicePidAccountingStats_v1
graphicsPidInfo
    dcgmGpuUsageInfo_t
graphicsProcesses
    dcgmDiagResponse_v3
groupCpuAffinityMask
    dcgmGroupTopology_v1
groupName
    dcgmGroupInfo_v1
    dcgmGroupInfo_v2

```

H

```

hasGlobalInfo
    dcgmIntrospectFullFieldsExecTime_v1
    dcgmIntrospectFullMemory_v1
health
    dcgmHealthResponse_v2
    dcgmPidSingleInfo_t
    dcgmHealthResponse_v1
    dcgmGpuUsageInfo_t
hMaxResolution
    dcgmDeviceFbcSessionInfo_v1
hResolution
    dcgmDeviceFbcSessionInfo_v1
    dcgmDeviceVgpuEncSessions_v1
hwDiagnosticReturn
    dcgmDiagResponsePerGpu_t

```

I

i64

- dcgmFieldValue_v1
- dcgmFieldValue_v2

id

dcgmModuleGetStatusesModule_t

identifiers

dcgmDeviceAttributes_v1

incidentCount

dcgmHealthResponse_v1

dcgmHealthResponse_v2

inforom

dcgmDiagResponse_v3

inforomImageVersion

dcgmDeviceIdentifiers_v1

introspectLvl

dcgmIntrospectContext_v1

isolation

dcgmPolicy_v1

K

kernel

dcgmIntrospectCpuUtil_v1

L

licenseStatus

dcgmVgpuInstanceAttributes_v1

linkState

dcgmNvLinkGpuLinkStatus_t

dcgmNvLinkNvSwitchLinkStatus_t

localNvLinkIds

dcgmDeviceTopology_v1

location

dcgmPolicyConditionDbe_t

lowUtilizationTime

dcgmPidSingleInfo_t

dcgmGpuUsageInfo_t

M

maxGpuMemoryUsed

dcgmPidSingleInfo_t

dcgmGpuUsageInfo_t

maxInstances
 dcgmDeviceVgpuTypeInfo_v1
maxMemoryUsage
 dcgmDevicePidAccountingStats_v1
maxPowerLimit
 dcgmDevicePowerLimits_v1
maxResolutionX
 dcgmDeviceVgpuTypeInfo_v1
maxResolutionY
 dcgmDeviceVgpuTypeInfo_v1
maxValue
 dcgmStatSummaryInt64_t
 dcgmStatSummaryInt32_t
 dcgmStatSummaryFp64_t
meanUpdateFreqUsec
 dcgmIntrospectFieldsExecTime_v1
memClock
 dcgmClockSet_v1
memCopyUtil
 dcgmVgpuDeviceAttributes_v5
memoryClock
 dcgmPidSingleInfo_t
 dcgmGpuUsageInfo_t
memoryUsage
 dcgmDeviceAttributes_v1
memoryUsed
 dcgmRunningProcess_v1
memoryUtilization
 dcgmPidSingleInfo_t
 dcgmDevicePidAccountingStats_v1
 dcgmGpuUsageInfo_t
memUtil
 dcgmDeviceVgpuUtilInfo_v1
 dcgmDeviceVgpuProcessUtilInfo_v1
minPowerLimit
 dcgmDevicePowerLimits_v1
minValue
 dcgmStatSummaryInt32_t
 dcgmStatSummaryFp64_t
 dcgmStatSummaryInt64_t
mode
 dcgmPolicy_v1

mpr
 dcgmPolicyCallbackResponse_v1

N

numaOptimalFlag
 dcgmGroupTopology_v1
numComputePids
 dcgmGpuUsageInfo_t
numDisplayHeads
 dcgmDeviceVgpuTypeInfo_v1
numerrors
 dcgmPolicyConditionDbe_t
numFieldIds
 dcgmFieldGroupInfo_v1
numGpus
 dcgmDeviceTopology_v1
 dcgmNvLinkStatus_v1
 dcgmPidInfo_v1
 dcgmJobInfo_v2
numGraphicsPids
 dcgmGpuUsageInfo_t
numNvSwitches
 dcgmNvLinkStatus_v1
numOtherComputePids
 dcgmPidSingleInfo_t
numOtherGraphicsPids
 dcgmPidSingleInfo_t
numXidCriticalErrors
 dcgmGpuUsageInfo_t
 dcgmPidSingleInfo_t
nvlink
 dcgmPolicyCallbackResponse_v1
nvmLibrary
 dcgmDiagResponse_v3
nvSwitches
 dcgmNvLinkStatus_v1

O

otherComputePids
 dcgmPidSingleInfo_t
otherGraphicsPids
 dcgmPidSingleInfo_t

overallHealth
 dcgmPidSingleInfo_t
 dcgmHealthResponse_v2
 dcgmHealthResponse_v1
 dcgmGpuUsageInfo_t

P

pageRetirement
 dcgmDiagResponse_v3

parms
 dcgmPolicy_v1

path
 dcgmDeviceTopology_v1

pci
 dcgmPolicyCallbackResponse_v1

pciBusId
 dcgmDeviceIdentifiers_v1

pciDeviceId
 dcgmDeviceIdentifiers_v1

pcieReplays
 dcgmGpuUsageInfo_t
 dcgmPidSingleInfo_t

pcieRxBandwidth
 dcgmPidSingleInfo_t
 dcgmGpuUsageInfo_t

pcieTxBandwidth
 dcgmPidSingleInfo_t
 dcgmGpuUsageInfo_t

pciSubSystemId
 dcgmDeviceIdentifiers_v1

perfState
 dcgmConfig_v1
 dcgmVgpuConfig_v1

perGpuResponses
 dcgmDiagResponse_v3

permissions
 dcgmDiagResponse_v3

persistAfterDisconnect
 dcgmConnectV2Params_v1
 dcgmConnectV2Params_v2

persistenceMode
 dcgmDiagResponse_v3

pid
dcgmDevicePidAccountingStats_v1
dcgmDeviceFbcSessionInfo_v1
dcgmDeviceVgpuEncSessions_v1
dcgmDeviceVgpuProcessUtilInfo_v1
dcgmPidInfo_v1
dcgmRunningProcess_v1

power
dcgmPolicyCallbackResponse_v1

powerLimit
dcgmVgpuConfig_v1
dcgmConfig_v1

powerLimits
dcgmDeviceAttributes_v1

powerUsage
dcgmGpuUsageInfo_t

powerViolation
dcgmPolicyConditionPower_t

powerViolationTime
dcgmGpuUsageInfo_t
dcgmPidSingleInfo_t

processName
dcgmDeviceVgpuProcessUtilInfo_v1

processUtilization
dcgmPidSingleInfo_t

R

recentUpdateUsec
dcgmIntrospectFieldsExecTime_v1

reliabilityViolationTime
dcgmPidSingleInfo_t
dcgmGpuUsageInfo_t

response
dcgmPolicy_v1

results
dcgmDiagResponsePerGpu_t

S

sbepages
dcgmPolicyConditionMpr_t

serial
dcgmDeviceIdentifiers_v1

sessionCount
 dcgmDeviceFbcStats_v1
 dcgmDeviceFbcSessions_v1
 dcgmDeviceEncStats_v1

sessionFlags
 dcgmDeviceFbcSessionInfo_v1

sessionId
 dcgmDeviceVgpuEncSessions_v1
 dcgmDeviceFbcSessionInfo_v1

sessionInfo
 dcgmDeviceFbcSessions_v1

sessionType
 dcgmDeviceFbcSessionInfo_v1

shutdownTemp
 dcgmDeviceThermals_v1

slowdownTemp
 dcgmDeviceThermals_v1

slowestPath
 dcgmGroupTopology_v1

smClock
 dcgmGpuUsageInfo_t
 dcgmClockSet_v1
 dcgmPidSingleInfo_t

smUtil
 dcgmDeviceVgpuUtilInfo_v1
 dcgmDeviceVgpuProcessUtilInfo_v1

smUtilization
 dcgmPidSingleInfo_t
 dcgmGpuUsageInfo_t

startTime
 dcgmPidSingleInfo_t
 dcgmGpuUsageInfo_t

startTimestamp
 dcgmDevicePidAccountingStats_v1

status
 dcgmModuleGetStatusesModule_t
 dcgmErrorInfo_t
 dcgmFieldValue_v1
 dcgmFieldValue_v2

str
 dcgmFieldValue_v2
 dcgmFieldValue_v1

subsystemId
`dcmDeviceVgpuTypeInfo_v1`

summary
`dcmJobInfo_v2`
`dcmPidInfo_v1`

supportedVgpuTypeCount
`dcmVgpuDeviceAttributes_v5`

supportedVgpuTypeInfo
`dcmVgpuDeviceAttributes_v5`

syncBoost
`dcmConfigPerfStateSettings_t`

syncBoostTime
`dcmPidSingleInfo_t`
`dcmGpuUsageInfo_t`

system
`dcmGpuUsageInfo_t`
`dcmPidSingleInfo_t`
`dcmHealthResponse_v1`
`dcmHealthResponse_v2`

systemError
`dcmDiagResponse_v3`

T

targetClocks
`dcmConfigPerfStateSettings_t`

thermal
`dcmPolicyCallbackResponse_v1`

thermalSettings
`dcmDeviceAttributes_v1`

thermalViolation
`dcmPolicyConditionThermal_t`

thermalViolationTime
`dcmPidSingleInfo_t`
`dcmGpuUsageInfo_t`

timeoutMs
`dcmConnectV2Params_v2`

timestamp
`dcmPolicyConditionDbe_t`
`dcmPolicyConditionPci_t`
`dcmPolicyConditionMpr_t`
`dcmPolicyConditionThermal_t`
`dcmPolicyConditionPower_t`
`dcmPolicyConditionNvlink_t`

```

dcgmPolicyConditionXID_t
total
    dcgmIntrospectCpuUtil_v1
totalEverUpdateUsec
    dcgmIntrospectFieldsExecTime_v1
ts
    dcgmFieldValue_v2
    dcgmFieldValue_v1
type
    dcgmConfigPowerLimit_t

```

U

```

unused
    dcgmFieldValue_v2
unusedActiveVgpuInstanceCount
    dcgmDeviceAttributes_v1
unusedCreatableVgpuTypeCount
    dcgmDeviceVgpuIds_v1
unusedCreatableVgpuTypeIds
    dcgmDeviceVgpuIds_v1
unusedSupportedVgpuTypeCount
    dcgmDeviceVgpuIds_v1
unusedSupportedVgpuTypeIds
    dcgmDeviceVgpuIds_v1
unusedVgpuIds
    dcgmDeviceAttributes_v1
unusedVgpuInstanceIds
    dcgmDeviceAttributes_v1
user
    dcgmIntrospectCpuUtil_v1
uuid
    dcgmDeviceIdentifiers_v1

```

V

```

val
    dcgmConfigPowerLimit_t
validation
    dcgmPolicy_v1
value
    dcgmFieldValue_v2
    dcgmFieldValue_v1
vbios
    dcgmDeviceIdentifiers_v1

```

version

dcmGroupInfo_v1
dcmDeviceIdentifiers_v1
dcmPolicy_v1
dcmPolicyCallbackResponse_v1
dcmDeviceMemoryUsage_v1
dcmFieldValue_v1
dcmFieldValue_v2
dcmGroupInfo_v2
dcmDeviceVgpuUtilInfo_v1
dcmHealthResponse_v1
dcmHealthResponse_v2
dcmDeviceEncStats_v1
dcmPidInfo_v1
dcmJobInfo_v2
dcmFieldGroupInfo_v1
dcmDeviceFbcStats_v1
dcmNvLinkStatus_v1
dcmIntrospectCpuUtil_v1
dcmRunningProcess_v1
dcmIntrospectMemory_v1
dcmIntrospectFullFieldsExecTime_v1
dcmIntrospectFieldsExecTime_v1
dcmIntrospectContext_v1
dcmDiagResponse_v3
dcmDeviceFbcSessionInfo_v1
dcmDeviceTopology_v1
dcmGroupTopology_v1
dcmClockSet_v1
dcmDeviceFbcSessions_v1
dcmDeviceVgpuEncSessions_v1
dcmConnectV2Params_v1
dcmDeviceSupportedClockSets_v1
dcmDeviceVgpuProcessUtilInfo_v1
dcmIntrospectFullMemory_v1
dcmDeviceVgpuIds_v1
dcmVgpuConfig_v1
dcmConfig_v1
dcmVgpuInstanceAttributes_v1
dcmVgpuDeviceAttributes_v5
dcmDevicePidAccountingStats_v1
dcmDeviceVgpuTypeInfo_v1
dcmDeviceAttributes_v1

```

dcgmConnectV2Params_v2
dcgmDeviceThermals_v1
dcgmDevicePowerLimits_v1
vgpuDriverVersion
    dcgmVgpuInstanceAttributes_v1
vgpuId
    dcgmDeviceVgpuEncSessions_v1
    dcgmDeviceVgpuProcessUtilInfo_v1
    dcgmDeviceVgpuUtilInfo_v1
    dcgmDeviceFbcSessionInfo_v1
vgpuProcessSamplesCount
    dcgmDeviceVgpuProcessUtilInfo_v1
vgpuTypeClass
    dcgmDeviceVgpuTypeInfo_v1
vgpuTypeId
    dcgmVgpuInstanceAttributes_v1
vgpuTypeInfo
    dcgmDeviceVgpuTypeInfo_v1
vgpuTypeLicense
    dcgmDeviceVgpuTypeInfo_v1
vgpuTypeName
    dcgmDeviceVgpuTypeInfo_v1
vgpuUtilInfo
    dcgmVgpuDeviceAttributes_v5
vgpuUuid
    dcgmVgpuInstanceAttributes_v1
violationOccurred
    dcgmPolicyViolationNotify_t
virtualizationMode
    dcgmDeviceIdentifiers_v1
vMaxResolution
    dcgmDeviceFbcSessionInfo_v1
vmId
    dcgmVgpuInstanceAttributes_v1
vmName
    dcgmVgpuInstanceAttributes_v1
vResolution
    dcgmDeviceFbcSessionInfo_v1
    dcgmDeviceVgpuEncSessions_v1

X
xid
    dcgmPolicyCallbackResponse_v1

```

xidCriticalErrorsTs

dcgmGpuUsageInfo_t
dcgmPidSingleInfo_t

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© 2013-2018 NVIDIA Corporation. All rights reserved.