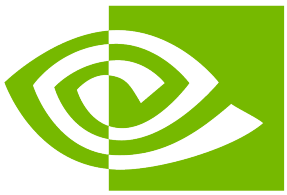

GPU Operational Events (Preview)

Release 615

NVIDIA Corporation

Sep 09, 2026

Contents

Warning

GPU Operational Events are a **preview feature** and remain under active development. Category values, module signatures, module event codes, severities, context payloads, and the set of published events may change before the interface is declared stable. Treat the identifiers in this document as valid for the driver release documented here, and re-validate consumers against each driver release. Legacy Xid output remains the stable contract for production monitoring during the preview period.

This document explains what GPU Operational Events are, how to monitor them, the formats they are published in, and the published event categories and codes. It is intended to assist system administrators, developers, and FAEs who consume structured GPU health and error data from the NVIDIA driver.

Chapter 1. Analyzing GPU Operational Events

Warning

GPU Operational Events are a **preview feature** and remain under active development. Category values, module signatures, module event codes, severities, context payloads, and the set of published events may change before the interface is declared stable. Treat the identifiers in this document as valid for the driver release documented here, and re-validate consumers against each driver release. Legacy Xid output remains the stable contract for production monitoring during the preview period.

1.1. Overview

A **GPU Operational Event (GOE)** is a structured, machine-readable record that the NVIDIA GPU driver publishes whenever it detects a condition worth reporting to a system operator: an error, a recovery action, a resource retirement, or a notable state change. Each event carries an identity, a severity, an impact scope, and a list of typed context payloads that describe the specific occurrence.

GPU Operational Events give system software a programmatic interface for the same conditions that the driver reports as Xid messages in the kernel log. Each event carries its identity in discrete numeric fields and provides higher attribution resolution than legacy Xid output.

Events that have a legacy Xid equivalent continue to produce their existing `NVRM: Xid (PCI:...)` kernel log lines unchanged. The structured event and the legacy Xid line describe the same occurrence.

For the actions to take in response to a condition, see the Xid Errors documentation. Each event in *Published Events* lists its legacy Xid, which selects the corresponding entry in the Xid catalog and its recommended immediate and investigatory actions.

1.2. Availability

GPU Operational Events are available on Linux through NVML, on Turing and later GPUs.

1.3. Event Identity

Event identity is consistent across all GPU Operational Event output formats, including structured records, CPER, and `nvidia-smi event-log`. Every GPU Operational Event is identified by a three-part tuple. A consumer uses the full tuple to recognize a specific event; individual parts are reused across events.

Field	Type	Description
Module Signature	Character array	NUL-terminated ASCII string naming the source module, for example GPU-GSP. Namespaces the event code.
Category	16-bit integer	Class of condition, from the category registry in *Event Categories*. Shared across all modules.
Module Event Code	16-bit integer	Specific event within the reporting module. Together with the module signature and category, selects an entry in *Published Events*.

Each event additionally carries the fields below.

Field	Description
Severity	Impact relative to the event scope. See *Event Severities*.
Log Level	Urgency of the report, orthogonal to severity. See *Event Log Levels*.
Scope	What the event affects. See *Event Scopes*.
Originator	Which driver or firmware component originated the event.
Device UUID	UUID of the GPU that reported the event.
Instance ID	Originator-scoped sequence number.
Timestamp	Microsecond wall clock at event creation.
Trace ID	Group identifier shared by related events that describe one incident.
Group Cursor	64-bit value that identifies the event group. Also maps to the CPER Error Record Header `recordId` for the retained CPER record.
Group Position	Index of this event within its group.
Attributes	Per-event flags that describe properties of the event.
Group Attributes	Group-level flags that describe properties of the event group.
Attribution	Finer-grained location of the event within the device. GPU Operational Events provide higher attribution resolution than legacy Xid messages.

Event records include grouping fields for correlating related events when the driver reports multiple events for one incident. In this release, published events are delivered individually. The trace ID identifies the incident. See *GPU Operational Event Formats* for how these fields appear in each published format.

1.4. Monitoring GPU Operational Events

The driver retains recent event groups in an in-memory Operational Event log and publishes them through the interfaces below. All interfaces describe the same events.

The log holds up to 64 event groups and evicts by severity. When the log is full, the driver evicts the oldest group at the lowest resident severity, and does so when the incoming group is at least as severe. When every resident group is more severe than the incoming group, the driver drops the incoming group and counts it as an overflow. Retained history therefore favors the most significant events over the most recent ones, so a long-running consumer should subscribe for live delivery and treat history replay as a catch-up mechanism.

1.4.1. Command Line

`nvidia-smi event-log` prints the events that have occurred since driver load in a human-readable form. It reports events for every GPU visible to NVML, skipping any GPU it cannot query. `event-log` accepts `-h` / `--help`. Resolve the numeric values it prints through *Published Events*. Three properties of the decoder in this release are worth noting when you do:

- Module event codes currently print as `Unknown` alongside the numeric value. The numeric value is correct and selects the catalog entry.
- Contexts other than Legacy Xid print their numeric type and no payload. Legacy Xid contexts print the Xid code.
- `Severity` reports the CPER severity of the record rather than the event's own severity, so a GPU-fatal event appears here as `Recoverable`. See *UEFI CPER* for the mapping, and read the severity in a structured record for GPU-relative impact.

`nvidia-smi cper` prints the same history as Base64-encoded raw CPER records, one record per Base64 block, for forwarding to an external collector. `cper` accepts `-h` / `--help` and `--no-wrap`. Output wraps at 76 columns by default; `--no-wrap` emits each CPER record on a single line.

1.4.2. NVML

NVML exposes GPU Operational Events through two interfaces. See the NVML API Reference Guide for full function prototypes.

Structured event subscription. Register a GPU with `nvmlEventSetRegisterGpuOperationalEvents_v1`, then retrieve events with `nvmlEventSetWait_v3`. Records with `dataType` set to `NVML_EVENT_DATA_TYPE_GPU_OPERATIONAL_EVENT` carry the structured event. Read attached context payloads with `nvmlEventSetGetContextCount_v1`, `nvmlEventSetGetContextInfo_v1`, and `nvmlEventSetGetContextData_v1`. Decode a legacy Xid context with `nvmlEventSetGetGpuOperationalEventContextLegacyXid_v1`.

The same event set can subscribe to legacy Xid events and GPU Operational Events. When a GPU Operational Event carries a Legacy Xid context, use that Xid value as a correlation hint to the matching Xid event, kernel log line, and catalog entry.

Retained CPER history. `nvmlSystemGetCPER_v1` returns retained CPER records as raw bytes. This is the interface that `nvidia-smi cper` uses.

1.4.3. Kernel Log

Events that carry a legacy Xid context emit their traditional kernel log line:

```
NVRM: Xid (PCI:0000:41:00 GPU-I:03): 63, Row Remapper: New row
↪(0x00000000fbc12000) marked for remapping, reset gpu to activate.
```

This format is unchanged from previous releases and remains the stable contract for existing log parsers. Events without a legacy Xid equivalent are visible through the structured, CPER, and out-of-band interfaces.

1.4.4. Out-of-Band

On platforms that support out-of-band reporting, the driver forwards the same rendered CPER record to the BMC, so a management controller observes the event without the host driver stack.

The category registry and the full list of published events for this driver release are in *GPU Operational Event Catalog*.

1.5. Interpreting an Event

1.5.1. Determine the Reporting Module and Event Code

Read the module signature and module event code from the structured record, or Source Module and Module Event Code from `nvidia-smi event-log`. Together with the category, this tuple selects an entry in *Published Events*, where events are grouped by module signature.

1.5.2. Determine Impact from the Category and Severity

The category identifies the class of condition and the severity identifies impact relative to the event's scope. Use scope to establish what the severity applies to: a fatal event at individual GPU engine scope reports that one engine is down, while a fatal event at device scope reports that the GPU is down.

1.5.3. Read the Context Payloads

Contexts carry occurrence-specific data. In this release, the Legacy Xid context is the one context that NVML and `nvidia-smi` decode into named fields: the Xid code and its formatted message. A consumer reads the remaining contexts as typed raw bytes and can skip any type it does not recognize using the size field.

1.5.4. Determine the Actions to Take

For an event that lists a legacy Xid in *Published Events*, look the Xid up in the Xid Errors catalog, which supplies the immediate and investigatory actions for that condition. Events that report a pending repair also carry the recovery action needed to activate the repair in their context payload.

Chapter 2. GPU Operational Event Formats

Warning

GPU Operational Events are a **preview feature** and remain under active development. Category values, module signatures, module event codes, severities, context payloads, and the set of published events may change before the interface is declared stable. Treat the identifiers in this document as valid for the driver release documented here, and re-validate consumers against each driver release. Legacy Xid output remains the stable contract for production monitoring during the preview period.

GPU Operational Events are published in several formats. Every format describes the same underlying events; consumers choose the format that fits their collection path.

Format	Typical Consumers
Structured Event Records	NVML subscribers and `nvidia-smi event-log`
UEFI CPER	Out-of-band BMC collection, `nvidia-smi cper`, and `nvmlSystemGetCPER_v1`
Legacy Xid Kernel Log Lines	Existing `dmesg` and syslog parsers

See *Analyzing GPU Operational Events* for how to monitor events and how to interpret categories and published codes.

2.1. Structured Event Records

Structured records expose the event identity and attribution as discrete fields. Through NVML, a successful `nvmlEventSetWait_v3` returns records whose `dataType` is `NVML_EVENT_DATA_TYPE_GPU_OPERATIONAL_EVENT`. Those records carry:

- Module signature, category, and module event code
- Severity, log level, scope, and originator
- Device UUID and attribution

- Trace ID, group cursor, group position, and instance ID
- Timestamp and per-event / group attributes

Context payloads follow the event. Each context has a type and a size. Bit 15 of the type partitions the space: types with bit 15 clear are domain-independent encodings, and types with bit 15 set are specific to the GPU domain.

Domain-independent types carry generically decodable data:

Type	Contents
`0x0000`	Opaque bytes
`0x0001`	Key/value pairs, 64-bit values
`0x0002`	Key/value pairs, 32-bit values
`0x0003`	Value array, 64-bit values
`0x0004`	Value array, 32-bit values

GPU domain types begin at `0x8001`; `0x8000` is reserved. The types defined in this release are:

Type	Name	Contents
`0x8001`	Legacy Xid	Legacy Xid code and formatted message
`0x9001`	GPU Timeout Data	Timeout-specific fields such as waited duration
`0xA001`	GPU Initialization Metadata	Driver and device metadata reported at initialization
`0xA101`	GSP RPC Timeout Data	RPC function, name, sequence number, and wait duration
`0xA201`	FSP Boot Timeout Data	FSP boot polling state
`0xA202`	FSP Fuse Error Data	FSP fuse status
`0xA301`	NVLink ALI Training Failure Data	Affected link IDs and debug registers
`0xA302`	NVLink MSE Error Data	MSE error status and debug registers
`0xA303`	NVLink Software Link-Down Data	Link ID of the link that went down

Types are allocated to reporting modules in blocks: `0xA000–0xA0FF` for events common to the GPU, `0xA100–0xA1FF` for GSP, `0xA200–0xA2FF` for FSP, `0xA300–0xA3FF` for NVLink, and `0xA400–0xA4FF` for Robust Channel. A consumer that does not recognize a context type skips it using the size field, which keeps a consumer working across releases that add types.

Legacy Xid is the one context type that NVML decodes into named fields, through `nvmlEventSetGetGpuOperationalEventContextLegacyXid_v1`. NVML reports every other context through its raw-data interfaces.

`nvidia-smi event-log` renders the same structured fields in human-readable form.

2.2. UEFI CPER

GPU Operational Events are encoded as UEFI Common Platform Error Records. See Appendix N of the UEFI Specification for the CPER format. CPER is the format used for out-of-band delivery to a BMC and for retained history access through `nvmlSystemGetCPER_v1` and `nvidia-smi cper`.

One CPER record represents one event group. Each event in the group becomes one section of type **NVIDIA Event Section**, GUID `9068e568-6ca0-11f0-aeaf-159343591eac`. The Creator ID and Notification Type GUIDs listed below are version-5 GUIDs derived from the NVIDIA namespace `ea219640-18b9-4c91-8e5c-58eeacd9b33f`. Match these GUIDs by value.

The NVIDIA Event Section begins with a 32-byte common header, followed by a device-type-specific event info structure, followed by the context list. The event identity maps as follows:

CPER Field	Source	Notes
Source Device Type	GPU (1)	Selects the event info layout.
Event Type (u16)	Category	Value from the category registry.
Event Sub-type (u16)	Module Event Code	Specific event within the reporting module.
Source Module Signature (`char[16]`)	Module Signature	NUL-terminated ASCII string naming the source module.
Event Trace ID (u64)	Trace ID	Full 64-bit value including the originator prefix.
Error Record Header `recordId`	Group Cursor	The driver writes the group cursor into `recordId`, so the same value identifies the event group in both the structured record and the CPER record.

Event info for the GPU device type adds the event originator, the device PDI, and the source partition and subpartition that locate the event within the device. Each context is emitted with a 16-byte header and padded to 16-byte alignment; context format types match the structured record context types, so a legacy Xid context appears as CPER context format type `0x8001`.

The Error Record Header identifies the producer through a **Creator ID** and describes how the condition was discovered through a **Notification Type**:

Role	GUID	Meaning
Creator ID	7d3f1537-bfc6-583a-997b-a7	Record created by GSP firmware on the physical function.
Creator ID	5a572ae4-9a04-5e55-a86f-db	Record created by the host driver on the physical function.
Notification Type	0eeaf365-3740-5271-ae58-10	Reported through a GPU hardware interrupt.
Notification Type	e0fd0b2d-ba62-59c1-9744-9d	Detected by on-device firmware fault handling.
Notification Type	3a90eac6-481d-5cd7-b57d-0a	Inferred from a timeout waiting on GPU hardware or firmware.
Notification Type	cb2668a9-0508-5e36-bbc9-4d	Detected by a software validation or invariant check.

GOE Severity to CPER Severity Mapping

GOE severity is evaluated at the event's explicit scope, such as a device or engine. CPER severity uses the UEFI encoding — recoverable (0), fatal (1), corrected (2), informational (3) — and is always evaluated at the host scope. Encoding a GOE as CPER therefore re-scopes the impact to the host, so the CPER severity can differ from the GOE severity.

The driver maps GOE fatal and GOE recoverable to CPER recoverable, GOE corrected to CPER corrected, and GOE informational to CPER informational. GPU-fatal conditions leave the host recoverable, so the driver emits CPER recoverable for them and does not emit CPER fatal. The NVIDIA Event Section carries identity and attribution fields, while GOE severity and scope remain on the structured event record. For GPU-relative impact, read those fields from the structured record; they are not recoverable from the CPER severity alone.

Group attributes map to Error Record Header flags: recovered, previous error, and simulated.

2.3. Legacy Xid Kernel Log Lines

Contexts of type `0x8001` carry a legacy Xid code and its formatted message. These produce the traditional kernel log line:

```
NVRM: Xid (PCI:0000:41:00 GPU-I:03): 63, Row Remapper: New row
↪(0x00000000fbc12000) marked for remapping, reset gpu to activate.
```

Most events that report an Xid also deliver that code and message as a `0x8001` context inside structured records and CPER, so a programmatic consumer may read them without parsing `dmesg`. The exception are some NVLink events in this release, which emit two separate GPU Operational Events for a single underlying occurrence: one that contains the event data formatted as an Xid context, and one that contains the event data as a structured NVLink event context. In that case, only one Xid line is emitted to kernel logs.

Chapter 3. GPU Operational Event Catalog

Warning

GPU Operational Events are a **preview feature** and remain under active development. Category values, module signatures, module event codes, severities, context payloads, and the set of published events may change before the interface is declared stable. Treat the identifiers in this document as valid for the driver release documented here, and re-validate consumers against each driver release. Legacy Xid output remains the stable contract for production monitoring during the preview period.

This page lists the shared event category registry and the GPU Operational Events published in this driver release. See *Analyzing GPU Operational Events* for monitoring interfaces and how to interpret an event, and *GPU Operational Event Formats* for the formats events are published in.

3.1. Event Categories

The category registry is shared by all reporting modules. A consumer can act on the category alone when it does not recognize a specific module event code.

Category	Value	Indicates
TIMEOUT	0x0001	An operation did not complete within its expected time budget.
MEMORY_INTEGRITY_ERROR	0x0002	A data integrity error in memory or cache, such as ECC, parity, or poison.
INTERCONNECT_ERROR	0x0003	An error on a link between the GPU and another device, such as NVLink, PCIe, or C2C.
CORRUPTION_DETECTED	0x0004	Corrupted driver or device state was detected.
SECURITY_ERROR	0x0005	A security policy violation or an authentication failure.
POWER_ERROR	0x0006	A power delivery or power management fault.
THERMAL_ERROR	0x0007	A thermal limit was reached or a thermal sensor failed.
FIRMWARE_FAULT	0x0008	A fault detected by firmware running on the device.
ENGINE_ERROR	0x0009	An error reported by a GPU engine.
RESOURCE_EXHAUSTED	0x000A	A repair, spare, or tracking resource has been used up.
TRANSLATION_FAULT	0x000B	An address translation or MMU fault.
UNCLASSIFIED_ERROR	0x7FFF	An error that has not yet been assigned a specific category.
INITIALIZATION	0x8000	Component initialization progress or completion.
SHUTDOWN	0x8001	Component shutdown or teardown.
STATE_CHANGED	0x8002	An operational state transition.
CONFIG_CHANGED	0x8003	A configuration change.
RESOURCE_RETIREMENT	0x8004	A resource has been marked for retirement or repair.
RESOURCE_THRESHOLD	0x8005	A resource usage threshold was crossed.
TELEMETRY	0xFFFE	Operational telemetry.
DEBUG	0xFFFF	Engineering and debug data.

This release publishes events in eight of these categories: TIMEOUT, MEMORY_INTEGRITY_ERROR, INTERCONNECT_ERROR, FIRMWARE_FAULT, RESOURCE_EXHAUSTED, UNCLASSIFIED_ERROR, INITIALIZATION, and RESOURCE_RETIREMENT. The remaining values are defined so that consumers can decode events introduced in later releases.

Note

In this release, the `nvidia-smi event-log` decoder renders category `0x0008` as "Engine Software Error". The numeric value is correct and corresponds to FIRMWARE_FAULT, so decode categories from the numeric value.

3.2. Event Log Levels

The log level describes the urgency of the report independently from the event severity. Values increase with urgency, so a subscription filter can use "at least this urgent" semantics. A filter value of 0 matches every log level.

Log Level	Value	Indicates
`TELEMETRY`	`10`	Operational telemetry.
`DIAGNOSTIC`	`20`	Diagnostic information. NVML names this level `NVML_GPU_OPERATIONAL_EVENT_LOG_LEVEL_DIAG`.
`NOTICE`	`30`	A notable condition that does not require immediate action.
`WARNING`	`40`	A condition that may require operator attention.
`ERROR`	`50`	A condition that requires operator attention.

3.3. Event Severities

The severity identifies the impact of the event relative to its scope. Values increase with severity, so a subscription filter can use "at least this severe" semantics. A filter value of 0 matches every severity.

Severity	Value	Indicates
`INFORMATIONAL`	`10`	The event reports state, telemetry, or progress.
`CORRECTED`	`20`	The condition was corrected without recovery action by the operator.
`RECOVERABLE`	`30`	The condition affected operation, and recovery is possible.
`FATAL`	`40`	The condition caused loss of the scoped resource.

When a GPU Operational Event is encoded as CPER, the GOE severity is remapped to a host-scoped CPER severity. See *GOE Severity to CPER Severity Mapping*.

3.4. Event Scopes

The scope identifies the resource affected by the event.

Scope	Value	Indicates
SYSTEM	0	The whole system.
DEVICE	1	One GPU device.
GPU_INSTANCE	2	One MIG GPU instance.
COMPUTE_INSTANCE	3	One MIG compute instance.
ENGINE	4	One GPU engine.

3.5. Published Events

This release publishes the events listed below. To identify an event you have decoded, find its module signature and module event code in the lookup table, then follow the event name to its entry. Not all Xids emitted by the driver have been fully converted to GPU Operational Events yet; more support will be added in future releases. To determine the actions to take, look up the event's legacy Xid in the Xid Errors catalog.

A severity listed as FATAL or RECOVERABLE is selected at runtime according to the impact of the reported condition.

3.5.1. Event Lookup

Module	Code	Event	Legacy Xid	Category
GPU	0x0001	*DRIVER_INIT*	None	INITIAL-IZATION
GPU-BUS	0x0002	*C2C_CONTAINMENT*	179	INTERCON-NECT_ERROR
GPU-FSP	0x0001	*BOOT_TIMEOUT*	143	TIMEOUT
GPU-FSP	0x0002	*FUSE_ERROR*	143	FIRMWARE_FAULT
GPU-GSP	0x0001	*RPC_TIMEOUT*	119	TIMEOUT
GPU-GSP	0x0002	*LIBOS_HEARTBEAT_TIMEOUT*	120	TIMEOUT
GPU-GSP	0x0003	*GSP_RM_HEARTBEAT_TIMEOUT*	120	TIMEOUT
GPU-GSP	0x0004	*FIRMWARE_FAULT*	120	FIRMWARE_FAULT
GPU-GSP	0x0005	*POISON*	140	MEM-ORY_INTEGRITY_ERROR
GPU-MEMSYS	0x0001	*MAINT_OP_ERROR*	175	MEM-ORY_INTEGRITY_ERROR
GPU-NVLINK	0x0001	*ALI_TRAINING_FAILURE*	136	INTERCON-NECT_ERROR

continues on next page

Table 1 – continued from previous page

Module	Code	Event	Legacy Xid	Category
GPU-NVLINK	0x0002	*MSE_DEGRADED*	150	FIRMWARE_FAULT
GPU-NVLINK	0x0003	*MSE_WATCHDOG_TIMEOUT*	150	FIRMWARE_FAULT
GPU-NVLINK	0x0004	*SW_LINK_DOWN*	155	INTERCONNECT_ERROR
GPU-RAS	0x0001	*ROW_REMAPPING_PENDING*	63	RESOURCE_RETIREMENT
GPU-RAS	0x0002	*ROW_REMAPPING_FAILURE_TABLE_FULL*	64	RESOURCE_EXHAUSTED
GPU-RAS	0x0003	*ROW_REMAPPING_FAILURE_BANK_FULL*	64	RESOURCE_EXHAUSTED
GPU-RAS	0x0004	*ROW_REMAPPING_FAILURE_INTERNAL_ERROR*	64	UNCLASSIFIED_ERROR
GPU-RAS	0x0005	*ROW_REMAPPING_FAILURE_RESERVED_ROW*	64	RESOURCE_EXHAUSTED
GPU-RAS	0x0006	*ROW_REMAPPING_FAILURE_PAGE_OFFLINE_F*	64	UNCLASSIFIED_ERROR
GPU-RAS	0x0007	*DRAM_RETIREMENT_FAILURE_INTERNAL_ERR*	64	UNCLASSIFIED_ERROR
GPU-RAS	0x0008	*DRAM_RETIREMENT_FAILURE_INFOROM_FULL*	64	RESOURCE_EXHAUSTED
GPU-RAS	0x0009	*DRAM_RETIREMENT_FAILURE_HW_LIMIT*	64	RESOURCE_EXHAUSTED
GPU-RAS	0x000A	*DRAM_RETIREMENT_FAILURE_NO_SPARE*	64	RESOURCE_EXHAUSTED
GPU-RAS	0x000B	*LTS_REPAIR_PENDING*	160	RESOURCE_RETIREMENT
GPU-RAS	0x000C	*LTS_REPAIR_FAILURE*	161	RESOURCE_EXHAUSTED
GPU-RAS	0x000D	*MEMORY_CHANNEL_REPAIR_PENDING*	160	RESOURCE_RETIREMENT
GPU-RAS	0x000E	*MEMORY_CHANNEL_REPAIR_FAILURE*	161	RESOURCE_EXHAUSTED
GPU-RAS	0x000F	*TPC_REPAIR_PENDING_SAME_GPC*	156	RESOURCE_RETIREMENT
GPU-RAS	0x0010	*TPC_REPAIR_PENDING_DIFFERENT_GPC*	156	RESOURCE_RETIREMENT
GPU-RAS	0x0011	*TPC_REPAIR_FAILURE_NO_SPARE*	157	RESOURCE_EXHAUSTED

continues on next page

Table 1 – continued from previous page

Module	Code	Event	Legacy Xid	Category
GPU-RAS	0x0012	*TPC_REPAIR_FAILURE_NO_SPARE_MIG*	157	RESOURCE_EXHAUSTED
GPU-RAS	0x0013	*ECC_RESIDUAL_UNCORRECTABLE_ERROR*	140	MEMORY_INTEGRITY_ERROR
GPU-RAS	0x0014	*DRAM_ECC_INTR_STORM*	92	MEMORY_INTEGRITY_ERROR
GPU-RAS	0x0015	*SM_ECC_INTR_STORM*	92	MEMORY_INTEGRITY_ERROR
GPU-RAS	0x0016	*BANK_REMAPPING_PENDING*	177	RESOURCE_RETIREMENT
GPU-RC	0x0001	*CHANNEL_RESET*	Varies	UNCLASSIFIED_ERROR

3.5.2. Event Details

Entries are grouped by source module signature, then ordered by category value and module event code.

3.5.2.1 GPU

3.5.2.1.1 DRIVER_INIT

Code
0x0001

Category
INITIALIZATION

Severity
INFORMATIONAL

Scope
DEVICE

Legacy Xid
None

GPU driver initialization completed for the device. Carries driver and device metadata.

3.5.2.2 GPU-BUS

3.5.2.2.1 C2C_CONTAINMENT

Code
0x0002

Category
INTERCONNECT_ERROR

Severity
FATAL or RECOVERABLE

Scope
DEVICE

Legacy Xid
179

C2C containment was triggered on the chip-to-chip link to the host CPU. Carries the containment code.

3.5.2.3 GPU-FSP

3.5.2.3.1 BOOT_TIMEOUT

Code
0x0001

Category
TIMEOUT

Severity
FATAL

Scope
DEVICE

Legacy Xid
143

The driver timed out polling for FSP boot completion during GPU initialization.

3.5.2.3.2 FUSE_ERROR

Code
0x0002

Category
FIRMWARE_FAULT

Severity
FATAL

Scope
DEVICE

Legacy Xid
143

The FSP fuse error check failed. Carries the fuse status.

3.5.2.4 GPU-GSP

3.5.2.4.1 RPC_TIMEOUT

Code
0x0001

Category
TIMEOUT

Severity
FATAL or RECOVERABLE

Scope
DEVICE

Legacy Xid
119

The driver timed out waiting for an RPC response from GSP. Carries the RPC function, name, sequence number, and wait duration.

3.5.2.4.2 LIBOS_HEARTBEAT_TIMEOUT

Code
0x0002

Category
TIMEOUT

Severity
FATAL or RECOVERABLE

Scope
DEVICE

Legacy Xid
120

The LibOS heartbeat from the GSP microcontroller stopped.

3.5.2.4.3 GSP_RM_HEARTBEAT_TIMEOUT

Code
0x0003

Category
TIMEOUT

Severity
FATAL or RECOVERABLE

Scope
DEVICE

Legacy Xid
120

The GSP-RM heartbeat stopped.

3.5.2.4.4 POISON

Code
0x0005

Category
MEMORY_INTEGRITY_ERROR

Severity
FATAL

Scope
DEVICE

Legacy Xid
140

GSP consumed poisoned data.

3.5.2.4.5 FIRMWARE_FAULT

Code
0x0004

Category
FIRMWARE_FAULT

Severity
FATAL

Scope
DEVICE

Legacy Xid
120

GSP firmware reported a fault.

3.5.2.5 GPU-MEMSYS

3.5.2.5.1 MAINT_OP_ERROR

Code
0x0001

Category
MEMORY_INTEGRITY_ERROR

Severity
FATAL

Scope
DEVICE

Legacy Xid
175

A GPU memory subsystem maintenance operation timed out. Carries the operation type.

3.5.2.6 GPU-NVLINK

3.5.2.6.1 ALI_TRAINING_FAILURE

Code
0x0001

Category
INTERCONNECT_ERROR

Severity
FATAL

Scope
DEVICE

Legacy Xid
136

NVLink ALI link training failed. Carries the affected link IDs and debug registers.

3.5.2.6.2 SW_LINK_DOWN

Code
0x0004

Category
INTERCONNECT_ERROR

Severity
FATAL

Scope
DEVICE

Legacy Xid
155

An NVLink went down while it was active. Carries the link ID.

3.5.2.6.3 MSE_DEGRADED

Code
0x0002

Category
FIRMWARE_FAULT

Severity
FATAL

Scope
DEVICE

Legacy Xid
150

The NVLink MSE transport entered a degraded state. The driver stops MSE communication until it is reloaded.

3.5.2.6.4 MSE_WATCHDOG_TIMEOUT

Code
0x0003

Category
FIRMWARE_FAULT

Severity
FATAL

Scope
DEVICE

Legacy Xid
150

The NVLink MSE watchdog expired. Carries the error status and debug registers.

3.5.2.7 GPU-RAS

3.5.2.7.1 ECC_RESIDUAL_UNCORRECTABLE_ERROR

Code
0x0013

Category
MEMORY_INTEGRITY_ERROR

Severity
FATAL or RECOVERABLE

Scope
DEVICE

Legacy Xid
140

An uncorrectable ECC error was detected outside the normal handling path. Carries DRAM, LTC, MMU, and PCIe error counts.

3.5.2.7.2 DRAM_ECC_INTR_STORM

Code
0x0014

Category
MEMORY_INTEGRITY_ERROR

Severity
CORRECTED

Scope
DEVICE

Legacy Xid
92

Single-bit ECC error interrupts were disabled on a framebuffer partition because of a high error rate. Carries the partition index.

3.5.2.7.3 SM_ECC_INTR_STORM

Code
0x0015

Category
MEMORY_INTEGRITY_ERROR

Severity
CORRECTED

Scope
DEVICE

Legacy Xid
92

An SM single-bit ECC interrupt storm was detected.

3.5.2.7.4 ROW_REMAPPING_FAILURE_TABLE_FULL

Code
0x0002

Category
RESOURCE_EXHAUSTED

Severity
FATAL

Scope
DEVICE

Legacy Xid
64

Row remapping failed because the row remapping table is full.

3.5.2.7.5 ROW_REMAPPING_FAILURE_BANK_FULL

Code
0x0003

Category
RESOURCE_EXHAUSTED

Severity
FATAL

Scope
DEVICE

Legacy Xid
64

Row remapping failed because all reserved rows for the bank are already remapped.

3.5.2.7.6 ROW_REMAPPING_FAILURE_RESERVED_ROW

Code
0x0005

Category
RESOURCE_EXHAUSTED

Severity
FATAL

Scope
DEVICE

Legacy Xid
64

Row remapping failed because the target is a reserved row.

3.5.2.7.7 DRAM_RETIREMENT_FAILURE_INFOROM_FULL

Code
0x0008

Category
RESOURCE_EXHAUSTED

Severity
FATAL

Scope
DEVICE

Legacy Xid
64

DRAM retirement failed because the InfoROM object is full.

3.5.2.7.8 DRAM_RETIREMENT_FAILURE_HW_LIMIT

Code
0x0009

Category
RESOURCE_EXHAUSTED

Severity
FATAL

Scope
DEVICE

Legacy Xid
64

DRAM retirement failed because a hardware limit was reached.

3.5.2.7.9 DRAM_RETIREMENT_FAILURE_NO_SPARE

Code
0x000A

Category
RESOURCE_EXHAUSTED

Severity
FATAL

Scope
DEVICE

Legacy Xid
64

DRAM retirement failed because no spare remains for retirement.

3.5.2.7.10 LTS_REPAIR_FAILURE

Code
0x000C

Category
RESOURCE_EXHAUSTED

Severity
FATAL

Scope
DEVICE

Legacy Xid
161

L2 slice repair failed because no spare L2 slices remain.

3.5.2.7.11 MEMORY_CHANNEL_REPAIR_FAILURE

Code
0x000E

Category
RESOURCE_EXHAUSTED

Severity
FATAL

Scope
DEVICE

Legacy Xid
161

Memory channel repair failed because no spare channels remain.

3.5.2.7.12 TPC_REPAIR_FAILURE_NO_SPARE

Code
0x0011

Category
RESOURCE_EXHAUSTED

Severity
FATAL

Scope
DEVICE

Legacy Xid
157

TPC retirement failed because no spare TPCs are available.

3.5.2.7.13 TPC_REPAIR_FAILURE_NO_SPARE_MIG

Code
0x0012

Category
RESOURCE_EXHAUSTED

Severity
FATAL

Scope
DEVICE

Legacy Xid
157

TPC retirement failed in MIG mode because no spare TPCs remain in the same GPC.

3.5.2.7.14 ROW_REMAPPING_FAILURE_INTERNAL_ERROR

Code
0x0004

Category
UNCLASSIFIED_ERROR

Severity
FATAL

Scope
DEVICE

Legacy Xid
64

Row remapping failed due to an internal driver error. Carries an error string.

3.5.2.7.15 ROW_REMAPPING_FAILURE_PAGE_OFFLINE_FAILURE

Code
0x0006

Category
UNCLASSIFIED_ERROR

Severity
FATAL

Scope
DEVICE

Legacy Xid
64

The row is already pending remapping. The remap occurs on the next GPU reset.

3.5.2.7.16 DRAM_RETIREMENT_FAILURE_INTERNAL_ERROR

Code
0x0007

Category
UNCLASSIFIED_ERROR

Severity
FATAL

Scope
DEVICE

Legacy Xid
64

DRAM retirement failed due to an internal driver error. The context identifies the specific reason.

3.5.2.7.17 ROW_REMAPPING_PENDING

Code
0x0001

Category
RESOURCE_RETIREMENT

Severity
RECOVERABLE

Scope
DEVICE

Legacy Xid
63

A new DRAM row was marked for remapping. The remap activates on the next GPU reset.

3.5.2.7.18 LTS_REPAIR_PENDING

Code
0x000B

Category
RESOURCE_RETIREMENT

Severity
RECOVERABLE

Scope
DEVICE

Legacy Xid
160

An L2 slice and its pair were marked for repair. Carries the location and the recovery action needed to activate the repair.

3.5.2.7.19 MEMORY_CHANNEL_REPAIR_PENDING

Code
0x000D

Category
RESOURCE_RETIREMENT

Severity
RECOVERABLE

Scope
DEVICE

Legacy Xid
160

A memory channel and its pair were marked for repair. Carries the location and the recovery action needed to activate the repair.

3.5.2.7.20 TPC_REPAIR_PENDING_SAME_GPC

Code
0x000F

Category
RESOURCE_RETIREMENT

Severity
RECOVERABLE

Scope
DEVICE

Legacy Xid
156

A TPC was retired and replaced with a spare from the same GPC.

3.5.2.7.21 TPC_REPAIR_PENDING_DIFFERENT_GPC

Code
0x0010

Category
RESOURCE_RETIREMENT

Severity
RECOVERABLE

Scope
DEVICE

Legacy Xid
156

A TPC was retired and replaced with a TPC from a different GPC.

3.5.2.7.22 BANK_REMAPPING_PENDING

Code
0x0016

Category
RESOURCE_RETIREMENT

Severity
RECOVERABLE

Scope
DEVICE

Legacy Xid
177

A new DRAM bank was marked for remapping. The remap activates on the next GPU reset.

3.5.2.8 GPU-RC

3.5.2.8.1 CHANNEL_RESET

Code
0x0001

Category
UNCLASSIFIED_ERROR

Severity
RECOVERABLE

Scope
ENGINE

Legacy Xid
Varies

Robust Channel recovery reset a channel or engine. The legacy Xid context carries the originating robust-channel Xid; the event context carries the root-cause Xid, faulting engine type, channel, TSG, runlist, and instance block.

Chapter 4. Notices

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

4.1. Trademarks

Copyright