



# COMPUTE SANITIZER

v2022.1.0 | December 2021

## Release Notes



# TABLE OF CONTENTS

<b>Chapter 1. Release Notes.....</b>	<b>1</b>
1.1. Updates in 2022.1.....	1
1.2. Updates in 2021.3.1.....	1
1.3. Updates in 2021.3.....	1
1.4. Updates in 2021.2.1.....	2
1.5. Updates in 2021.2.....	2
1.6. Updates in 2021.1.1.....	2
1.7. Updates in 2021.1.....	2
1.8. Updates in 2020.3.1.....	2
1.9. Updates in 2020.3.....	3
1.10. Updates in 2020.2.1.....	3
1.11. Updates in 2020.2.....	3
1.12. Updates in 2020.1.2.....	3
1.13. Updates in 2020.1.1.....	3
1.14. Updates in 2020.1.....	3
1.15. Updates in 2019.1.....	4
<b>Chapter 2. Known Limitations.....</b>	<b>5</b>
<b>Chapter 3. Known Issues.....</b>	<b>6</b>
<b>Chapter 4. Support.....</b>	<b>7</b>
4.1. Platform Support.....	7
4.2. GPU Support.....	7

## LIST OF TABLES

Table 1	Platforms supported by Compute Sanitizer .....	7
Table 2	GPU architectures supported by Compute Sanitizer .....	7



# Chapter 1.

## RELEASE NOTES

### 1.1. Updates in 2022.1

- ▶ Add support for generating coredumps.
- ▶ Improve support for stack overflow detection.
- ▶ Added new option `--target-processes-filter` to filter the processes being tracked by name.
- ▶ Initcheck support for asynchronous allocations. Requires CUDA driver version 510 or newer.
- ▶ Initcheck support for accesses on peer devices. Requires CUDA driver version 510 or newer.
- ▶ Added support for OptiX 7 applications.
- ▶ Added support for tracking the child processes of 32-bit processes in multi-process applications on Linux and Windows x86\_64.

### 1.2. Updates in 2021.3.1

- ▶ Fix intermittent issue on vGPU where synccheck would incorrectly detect divergent threads.
- ▶ Fix potential hang when tracking several graph launches.

### 1.3. Updates in 2021.3

- ▶ Improved Linux host backtrace.
- ▶ Remove requirement to call `cudaDeviceReset()` for accurate reporting of memory leaks and unused memory features.
- ▶ Fix synccheck potential hang when calling `__syncthreads` in divergent code paths on Volta GPUs or newer.
- ▶ Print nearest allocation information for memcheck precise errors in global memory.
- ▶ Add warning when calling device-side `malloc` with an empty size.

- ▶ Add separate public API device callback for `cuda::memcpy_async`.
- ▶ Add new command-line option `--num-cuda-barriers` to override the expected number of `cuda::barrier` used by the target application.
- ▶ Added new command-line options `--print-session-details` to print session information and `--save-session-details` to save it to the output file.
- ▶ Added support for WSL2.

## 1.4. Updates in 2021.2.1

- ▶ Add device backtrace for malloc/free errors in CUDA kernels.
- ▶ Improve racecheck host memory footprint.

## 1.5. Updates in 2021.2

- ▶ Added racecheck and synccheck support for `cuda::barrier` on Ampere GPUs or newer.
- ▶ Added racecheck support for `__syncwarp` with partial mask.
- ▶ Added `--launch-count` and `--launch-skip` filtering options. See the [Command Line Options documentation](#) for more information.
- ▶ `--filter` and `--exclude` options have been respectively renamed to `--kernel-regex` and `--kernel-regex-exclude`.
- ▶ Added support for QNX and Linux aarch64 platforms.
- ▶ Added support for CUDA graphs memory nodes.

## 1.6. Updates in 2021.1.1

- ▶ Fixed an issue where incorrect line numbers could be shown in errors reports.

## 1.7. Updates in 2021.1

- ▶ Support for allocation padding via the `--padding` option.
- ▶ Experimental support for NVTX memory API using option `--nvtx yes`. Please refer to [NVTX API for Compute Sanitizer Reference Manual](#) for more information.

## 1.8. Updates in 2020.3.1

- ▶ Fixed issue when launching a CUDA graph multiple times.
- ▶ Fixed false positives when using cooperative groups synchronization primitives with `initcheck` and `synccheck`.

## 1.9. Updates in 2020.3

- ▶ Added support for CUDA memory pools and CUDA API reduced serialization.
- ▶ Added host backtrace for unused memory reports.

## 1.10. Updates in 2020.2.1

- ▶ Fixed crash when loading cubins of size larger than 2 GiB.
- ▶ Fix error detection on systems with multiple GPUs.
- ▶ Fixed issue when using CUDA Virtual Memory Management API `cuMemSetAccess` to remove access to a subset of devices on a system with multiple GPUs.
- ▶ Added public API to translate between sanitizer and CUDA stream handles.

## 1.11. Updates in 2020.2

- ▶ Added support for CUDA graphs and CUDA memmap APIs.
- ▶ The memory access callback of the public API has been split into three distinct callbacks corresponding to global, shared and local memory accesses.

## 1.12. Updates in 2020.1.2

- ▶ Added sanitizer stream API. This fixes tool crashes when per-thread streams are being used.

## 1.13. Updates in 2020.1.1

- ▶ Support for Windows Hardware-accelerated GPU scheduling
- ▶ Support for tracking child processes spawned by the application launched under the tool via the `--target-processes` CLI option.

## 1.14. Updates in 2020.1

- ▶ Initial release of the Compute Sanitizer (with CUDA 11.0)

Updates to the Sanitizer API :

- ▶ Added support for per-thread streams
- ▶ Added APIs to retrieve the PC and size of a CUDA function or patch
- ▶ Added callback for `cudaStreamAttachMemAsync`
- ▶ Added direction to memcpy callback data
- ▶ Added stream to memcpy and memset callbacks data

- ▶ Added launch callback after syscall setup
- ▶ Added visibility field to allocation callback data
- ▶ Added PC argument to block entry callback
- ▶ Added incoming value to memory access callbacks
- ▶ Added threadCount to barrier callbacks
- ▶ Added cooperative group flags for barrier and function callbacks

## 1.15. Updates in 2019.1

- ▶ Initial release of the Compute Sanitizer API (with CUDA 10.1)



## Chapter 2.

# KNOWN LIMITATIONS

- ▶ Applications run much slower under the Compute Sanitizer tools. This may cause some kernel launches to fail with a launch timeout error when running with the Compute Sanitizer enabled.
- ▶ Compute Sanitizer tools do not support device backtrace on Maxwell devices (SM 5.x).
- ▶ Compute Sanitizer tools do not support device backtrace on Windows Server 2016 for devices in WDDM mode.
- ▶ Compute Sanitizer tools do not support device backtrace on WSL2.
- ▶ Compute Sanitizer tools do not support CUDA/Direct3D interop.
- ▶ Compute Sanitizer tools do not support CUDA/Vulkan interop.
- ▶ The memcheck tool does not support CUDA API error checking for API calls made on the GPU using dynamic parallelism.
- ▶ The racecheck, synccheck and initcheck tools do not support CUDA dynamic parallelism.
- ▶ CUDA dynamic parallelism is not supported when Windows Hardware-accelerated GPU scheduling is enabled.
- ▶ Compute Sanitizer tools cannot interoperate with other CUDA developer tools. This includes CUDA coredumps which are automatically disabled by the Compute Sanitizer. They can be enabled instead by using the `--generate-coredump` option.
- ▶ Compute Sanitizer tools do not support IPC memory pools. Using it will result in false positives.
- ▶ Compute Sanitizer tools are not supported when SLI is enabled.

## Chapter 3.

# KNOWN ISSUES

- ▶ The racecheck tool may print incorrect data for "Current value" when reporting a hazard on a shared memory location where the last access was an atomic operation. This can also impact the severity of this hazard.
- ▶ With some versions of Windows Server 2016, programs built with some configurations might hang when used with the Compute Sanitizer. A workaround for this issue is to use the Computer Sanitizer with **--show-backtrace device** or **--show-backtrace no** options.
- ▶ On QNX, when using the **--target-processes all** option, analyzing shell scripts may hang after the script has completed. End the application using *Ctrl-C* on the command line in that case.
- ▶ The initcheck tool might report false positives for device-to-host `cudaMemcpy` operations on padded structs that were initialized by a CUDA kernel. The **#pragma pack** directive can be used to disable the padding as a workaround.

# Chapter 4.

## SUPPORT

Information on supported platforms and GPUs.

### 4.1. Platform Support

Table 1 Platforms supported by Compute Sanitizer

Platform	Support
Windows	Yes
Linux (x86_64)	Yes
Linux (ppc64le)	Yes
Linux (aarch64bsa)	Yes
Linux (aarch64)	Yes
QNX	Yes
MacOSX	No

### 4.2. GPU Support

Table 2 GPU architectures supported by Compute Sanitizer

Architecture	Support
Kepler	No
Maxwell	Yes
Pascal	Yes
Volta	Yes
Turing	Yes
Ampere	Yes

## **Notice**

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 of 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.

## **Trademarks**

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.

## **Copyright**

© 2019-2021 NVIDIA Corporation and affiliates. All rights reserved.

This product includes software developed by the Syncro Soft SRL (<http://www.sync.ro/>).