NVIDIA Trusted Computing Solutions

Release Notes

RN-12523-001_r580_02 | August 2025

Document History

RN-12523-001_r580_02

Version	Date	Description of Change
02	August 2025	Updated the VBIOS component version to enable SPT CC mode for Blackwell GPUs
01	August 2025	R580 TRD1 release

Table of Contents

Overview

This release consists of the NVIDIA® CUDA® Toolkit version 13.0, which is paired with the NVIDIA Data Center GPU Drivers version 580.65.06.

The following features are supported in this software release:

- Protected PCIe (PPCIe) mode for NVIDIA Hopper GPUs
- Single GPU Passthrough (SPT) for NVIDIA Hopper GPUs
- Single GPU Passthrough (SPT) for NVIDIA Blackwell GPUs
- Single GPU Passthrough (SPT) for RTX PRO 6000 Blackwell Server Edition **NEW**

Refer to Feature Summary for more information.

Before you deploy workloads, NVIDIA recommends that users use good practices, such as performing regular attestations.

Feature Summary

Confidential Computing

This section provides information about the CC features in this release.

Hopper Single GPU Passthrough with a Bounce Buffer

NVIDIA® Trusted Computing support for NVIDIA Hopper™ GPUs was first introduced with the Hopper Single GPU Passthrough with a Bounce Buffer (SPT CC) mode. In this mode, one GPU can be passed through for each Confidential VM (CVM). A bounce buffer stages encrypted data transfers between the GPU device and the CVM. Refer to the *Intel TDX - Confidential Computing Deployment Guide and AMD SNP - Confidential Computing Deployment Guide* for more information.

This release introduces support for hosting multiple CVMs on the same node, each with one GPU passed through.

Table 1. Component Versions to Enable SPT CC Mode for Hopper GPUs

Component	Minimum Version
VBIOS	H100/H200: Hopper FW 1.8.0 [96.00.D9.00.XX] H20: 96.00.D0.00.06 H800: 96.00.5E.00.02
Host OS Kernel	Intel TDX Kernel 6.12 (Vendor fork) AMD SEV/SEV-SNP 6.11+
Guest OS	Ubuntu 24.04
`gpu_admin.py`	The main branch is github.com/nvidia/nvtrust. GPU tools - v2025.04.07
Attestation SDK Local GPU Verifier	Version 2.5.0 or later

Blackwell B200 Single GPU Passthrough with a Bounce Buffer

This release supports Blackwell Single GPU Passthrough with a Bounce Buffer (SPT CC) mode at a General Availability (GA) level. In this mode, one GPU can be passed through for each Confidential VM (CVM). A bounce buffer stages encrypted data transfers between the GPU device and CVM. Key Rotation is not currently supported in this mode for Blackwell.

Table 2. Component Versions to Enable SPT CC Mode for Blackwell GPUs

Component	Version
VBIOS	HGX Blackwell FW 1.2.0 [97.00.C5.00.xx]
Host OS	Intel TDX Kernel 6.12 (Vendor fork) AMD SEV/SEV-SNP 6.11
Guest OS	Ubuntu 24.04
`gpu_admin.py`	The main branch is github.com/nvidia/nvtrust. GPU tools - v2025.04.07
Attestation SDK Local GPU Verifier	Version 2.6.0 or later

RTX PRO 6000 Blackwell Server Edition Single GPU Passthrough with a Bounce Buffer

This release introduces SPT support for RTX PRO 6000 Blackwell Server Edition. In this mode, one GPU can be passed through for each Confidential VM (CVM). A bounce buffer stages encrypted data transfers between the GPU device and CVM. Transactions between the GPU and external video memory DRAM are encrypted. Key Rotation is not currently supported in this mode for Blackwell.

Table 3. Component Versions to Enable SPT CC Mode for Blackwell GPUs

Component	Version
VBIOS	98.02.81.00.01
Host OS	Intel TDX Kernel 6.12 (Vendor fork) AMD SEV/SEV-SNP 6.11+
Guest OS	Ubuntu 24.04
`gpu_admin.py`	The main branch is github.com/nvidia/nvtrust. GPU tools - v2025.04.07
Attestation SDK Local GPU Verifier	Version 2.6.0 or later

Protected PCIe

This section provides information about the PPCIe features in this release.

Eight Hopper GPUs with Four NVSwitch Passthrough

Trusted Computing support in the PPCIe mode is available with Hopper GPUs and Intel® CPUs with TDX/AMD CPUs with SEV/SEV-SNP technology in an Ubuntu KVM/QEMU environment.

In the PPCIe mode, multiple Hopper GPUs interconnected by NVSwitch or NVLink can be passed through to one CVM. As in the SPT CC mode, a bounce buffer is used to stage encrypted data transfers between the GPU device and CVM over the PCI Express bus. In this mode, GPU-GPU communications over the NVLink or NVSwitch interconnect are not encrypted.

PPCIe modes are only supported for the Hopper architecture.

Table 4. Component Versions to Enable PPCIe

Component	Minimum Version
HGX Hopper FW bundle	HGX Hopper FW 1.8.0 [96.00.D9.00.XX]
CVM Kernel	Intel TDX Kernel 6.12 (Vendor fork) AMD SEV/SEV-SNP 6.11+
Base OS	Ubuntu 24.04
gpu_admin.py	The main branch is github.com/nvidia/nvtrust. GPU tools - v2025.04.07
PPCIE Verifier	Version 1.5.0 or later

Limitations

This section provides a list of the known limitations in this release.

Limitations in the SPT CC Mode

- Only one GPU per CVM is allowed.
  This limitation is temporary and is expected to be resolved in a future release.
- With a maximum of one GPU passed through per CVM, operations that involve multiple GPUs, such as peer-to-peer communications, are not supported.

Limitations in the Hopper PPCIe Mode

- Hopper PPCIe is limited to HGX 8-way Air Cooled systems, where the eight Hopper GPUs and four NVSwitches are passed through to one VM.
  Other topologies are not supported.
- NVIDIA NCCL is the only supported GPU communication library.
- In Protected PCIe configs, P2P communication is only supported over NVLINK. P2P over PCI-Express is not supported.
- In the PPCIe mode, when the source or destination operand is imported, GPU memory allocations on a device that is not visible to the process, the host-to-device, or device-to-host copies might fail asynchronously with `cudaErrorLaunchFailure`.
- In the PPCIe mode, using `cooperative_groups::multi_grid_group::sync` in kernels launched with `cudaLaunchCooperativeKernelMultiDevice` results in the kernel failing with `cudaErrorIllegalAddress`.
- CUDA Interprocess Communication (IPC) is not supported in PPCIe mode.
- Developer tools such as NVIDIA Nsight for profiling are not supported in PPCIe mode
- The following driver multicast APIs are not supported:
  - `cuMulticastAddDevice`
  - `cuMulticastBindAddr`
  - `cuMulticastBindMem`
  - `cuMulticastCreate`
  - `cuMulticastGetGranularity`
  - `cuMulticastUnbind`

Limitations in the SPT CC and PPCIe Modes

This section provides information about the limitations that apply to both the SPT CC and PPCIe modes.

The following runtime APIs are incompatible with CC:

- Host memory registration
  The following CPU memory pinning operations are not allowed in CC mode:
  - `cudaHostRegister`
  - `cudaHostUnregister`
- `cudaMemcpy` calls that describe a Host-to-Array or Array-to-Host copy
  The following Host-to-Array and Array-to-Host copies are not supported because of the potential requirement for a conversion between pitch-linear and block-linear access patterns of the CUArray memory type during the secure copy operation:
  - `cudaMemcpy2DFromArray`
  - `cudaMemcpy2DFromArrayAsync`
  - `cudaMemcpy2DToArray`
  - `cudaMemcpy2DToArrayAsync`
  - `cudaMemcpy3D`
  - `cudaMemcpy3DAsync`
  - `cudaMemcpy3DPeer`
  - `cudaMemcpy3DPeerAsync`
- CUDA External Resource Interoperability
  The following APIs are not supported because an external resource interaction with a trusted execution environment is not permitted:
  - `cudaImportExternalMemory`
  - `cudaExternalMemoryGetMappedBuffer`
  - `cudaExternalMemoryGetMappedMipmappedArray`
  - `cudaDestroyExternalMemory`
  - `cudaFreeMipmappedArray`
  - `cudaImportExternalSemaphore`
  - `cudaSignalExternalSemaphoresAsync`
  - `cudaWaitExternalSemaphoresAsync`
  - `cudaDestroyExternalSemaphore`
  - `cudaGraphAddExternalSemaphoresSignalNode`
  - `cudaGraphAddExternalSemaphoresWaitNode`
  - `cudaGraphExecExternalSemaphoresSignalNodeSetParams`
  - cudaGraphExecExternalSemaphoresWaitNodeSetParams
  - `cudaGraphExternalSemaphoresSignalNodeGetParams`
  - `cudaGraphExternalSemaphoresSignalNodeSetParams`

- `cudaGraphExternalSemaphoresWaitNodeGetParams`
- `cudaGraphExternalSemaphoresWaitNodeSetParams`

The following CUDA Runtime APIs are not supported with CC in this release but might be enabled in a future release:

- `cudaEventElapsedTime`
- `cudaEventElapsedTime_v2`

The following Driver APIs are incompatible with CC:

- Host memory registration
  The following CPU memory pinning operations are not allowed in CC mode:
  - `cuMemHostRegister`
  - `cuMemHostUnregister`

- cuMemcpy calls that describe a Host-to-Array or Array-to-Host copy
  The following Host-to-Array and Array-to-Host copies are not supported because of the potential requirement for a conversion between pitch-linear and block-linear access patterns of the CUArray memory type during the secure copy operation:
  - `cuMemcpy2D`
  - `cuMemcpy2DAsync`
  - `cuMemcpy2DUnaligned`
  - `cuMemcpy3D`
  - `cuMemcpy3DAsync`
  - `cuMemcpyAtoH`
  - `cuMemcpyAtoHAsync`
  - `cuMemcpyHtoA`
  - `cuMemcpyHtoAAsync`

- cuStream memory operation calls passing pointers allocated using the `cudaMallocHost`, `cudaHostAlloc`, `cuMemAllocHost` APIs, and their graph counterparts:
  - `cuStreamBatchMemOp`
  - `cuStreamBatchMemOp_v2`
  - `cuStreamWaitValue32`
  - `cuStreamWaitValue32_v2`
  - `cuStreamWaitValue64`
  - `cuStreamWaitValue64_v2`
  - `cuStreamWriteValue32`
  - `cuStreamWriteValue32_v2`
  - `cuStreamWriteValue64`
  - `cuStreamWriteValue64_v2`
  - `cuGraphAddBatchMemOpNode`
  - `cuGraphBatchMemOpNodeGetParams`
  - `cuGraphBatchMemOpNodeSetParams`
  - `CuGraphExecBatchMemOpNodeSetParams`

- CUDA External Resource Interoperability
  The following APIs are not supported because external resource interaction with a trusted execution environment is not permitted:
  - `cuImportExternalMemory`
  - `cuExternalMemoryGetMappedBuffer`
  - `cuExternalMemoryGetMappedMipmappedArray`
  - `cuDestroyExternalMemory`
  - `cuFreeMipmappedArray`
  - `cuImportExternalSemaphore`
  - `cuSignalExternalSemaphoresAsync`
  - `cuWaitExternalSemaphoresAsync`
  - `cuDestroyExternalSemaphore`
  - `cuGraphAddExternalSemaphoresSignalNode`
  - `cuGraphAddExternalSemaphoresWaitNode`
  - `cuGraphExecExternalSemaphoresSignalNodeSetParams`
  - `cuGraphExecExternalSemaphoresWaitNodeSetParams`
  - `cuGraphExternalSemaphoresSignalNodeGetParams`
  - `cuGraphExternalSemaphoresSignalNodeSetParams`
  - `cuGraphExternalSemaphoresWaitNodeGetParams`
  - `cuGraphExternalSemaphoresWaitNodeSetParams`

The following CUDA driver APIs are not supported with CC in this release but might be enabled in a future release:

- `cuEventElapsedTime`
- `cuEventElapsedTime_v2`
- `cuMemBatchDecompressAsync`

The following CUDA capabilities are incompatible with CC:

- CUDA/Graphics interop, specifically APIs to enable interop with EGL, VDPAU, OpenGL, DirectX, OptiX, and Vulkan
- GPUDirect RDMA.
- The CUDA Programmatic Dependent Launch and Synchronization feature will not show expected overlaps in the primary and secondary kernel executions.
  A program that uses these APIs should functionally succeed in CC modes.

The following CUDA samples are expected to fail in the CC mode:

- `convolutionTexture`
- `dct8x8`
- `lineOfSight`
- `simpleCubemapTexture`
- `simpleLayeredTexture`
- `simplePitchLinearTexture`

- `simpleStream`
- `simpleTexture`
- `simpleTextureDrv`
- `watershedSegmentationNPP`
- `cudaCompressibleMemory`

The following CUDA samples are expected to fail in the PPCIe mode:

- `simpleIPC`
- `p2pBandwidthLatencyTest`

The following CUDA capabilities are not supported with CC in this release but might be enabled in a future release:

- CUDA Multi Process Service (MPS)
- CUDA Toolkit minor version compatibility
- CUDA Forward Compatibility

Known Issues

- A key rotation feature is not supported with PPCIe.
  A sophisticated attacker with physical or logical superuser access to the system can act as a passive adversary to capture the ciphertext and execute an attempt to break it or the key.

  Workaround
  Users should review the latest research on the effects of extreme AES key usage and the cryptographic wear out to determine their requirements for an attacker advantage. To create a new set of encryption keys in PPCIe mode, users must terminate and launch their CVMs again.

- In systems with multiple GPUs interconnected by NVSWITCH interconnects, the driver registry key `NVreg_RegistryDwords="RmNvswitchGpioDetect=2"` must be utilized if any of the virtual machines has only a single GPU passed through with CC-mode set to OFF.

  Workaround
  None

- With Blackwell SPT, there are restrictions in the number of decoder sessions.
  Due to a software bug, users cannot use more than 190 sessions when a single CUDA context is shared among all decoder sessions or no more than 28 sessions when one CUDA context per decoder session is created.

  Workaround
  None. This issue will be fixed in a future release.

- Performance is degraded with NCCL 2.26.2 and Protected PCIe.
  Applications might encounter up to 10% slowdown with NCCL 2.26.2.

  Workaround
  Use NCCL 2.26.3 or a newer version with the fix.

- IV exhaustion will crash the application in PPCIe mode.
  The H100 CC modes use a 96-bit deterministic IV for each virtual copy engine that is used to transfer data between the GPU and CPU. When this IV space is exhausted, transfers will fail to complete.

  Workaround
  Rotate the keys often in supported modes. If the keys are not rotated often, restart the CVM.

- The GPU-Ready bit is set when devtools mode is enabled.

  Workaround
  When in full CC-on modes, the driver will not accept any workloads until after the Attestation SDK, or the users, manually enable a GPU-Ready bit.

  > **Note:** This bit is already enabled in devtools mode.

  Users should use best practices by attesting the GPU before performing any work. The GPUs booted in devtools mode will be clearly identified, and the attestation will fail.

- With HGX Hopper Firmware 1.6.0, there is an increased risk of the GPU or NVSwitch falling off the PCIe bus during DC power cycling. This issue was resolved in the 1.7.0 firmware release.

  Workaround
  A system reboot would need to be performed to bring the missing devices back on the PCIe bus. Recommend using HGX Hopper Firmware 1.8.0 or newer.

- NVIDIA Performance Primitives (NPP) might not work.
  NPP uses optimized coding to extract the maximum performance from commonly used transforms and calculations as part of the leverage pinned host memory, which is not supported in CC.