

[NCCL](../index.html)

[2.25](https://docs.nvidia.com/deeplearning/sdk/nccl-archived/index.html)

- * [Overview of NCCL](../overview.html)

- * [Setup](../setup.html)

- * [Using NCCL](../usage.html)

- * [Creating a Communicator](../usage/communicators.html)

- * [Creating a communicator with options](../usage/communicators.html#creating-a-communicator-with-options)

- * [Creating a communicator using multiple ncclUniqueIds](../usage/communicators.html#creating-a-communicator-using-multiple-nccluniqueids)

- * [Creating more communicators](../usage/communicators.html#creating-more-communicators)

- * [Using multiple NCCL communicators concurrently](../usage/communicators.html#using-multiple-nccl-communicators-concurrently)

- * [Finalizing a communicator](../usage/communicators.html#finalizing-a-communicator)

- * [Destroying a communicator](../usage/communicators.html#destroying-a-communicator)

- * [Error handling and communicator abort](../usage/communicators.html#error-handling-and-communicator-abort)

- * [Asynchronous errors and error handling](../usage/communicators.html#asynchronous-errors-and-error-handling)

- * [Fault Tolerance](../usage/communicators.html#fault-tolerance)

- * [Collective Operations](../usage/collectives.html)

- * [AllReduce](../usage/collectives.html#allreduce)

- * [Broadcast](../usage/collectives.html#broadcast)

- * [Reduce](../usage/collectives.html#reduce)

- * [\[AllGather\]\(../usage/collectives.html#allgather\)](#)
- * [\[ReduceScatter\]\(../usage/collectives.html#reducescatter\)](#)
- * [\[Data Pointers\]\(../usage/data.html\)](#)
- * [\[CUDA Stream Semantics\]\(../usage/streams.html\)](#)
 - * [\[Mixing Multiple Streams within the same ncclGroupStart/End\(\) group\]\(../usage/streams.html#mixing-multiple-streams-within-the-same-ncclgroupstart-end-group\)](#)
- * [\[Group Calls\]\(../usage/groups.html\)](#)
 - * [\[Management Of Multiple GPUs From One Thread\]\(../usage/groups.html#management-of-multiple-gpus-from-one-thread\)](#)
 - * [\[Aggregated Operations \(2.2 and later\)\]\(../usage/groups.html#aggregated-operations-2-2-and-later\)](#)
 - * [\[Nonblocking Group Operation\]\(../usage/groups.html#nonblocking-group-operation\)](#)
- * [\[Point-to-point communication\]\(../usage/p2p.html\)](#)
 - * [\[Sendrecv\]\(../usage/p2p.html#sendrecv\)](#)
 - * [\[One-to-all \(scatter\)\]\(../usage/p2p.html#one-to-all-scatter\)](#)
 - * [\[All-to-one \(gather\)\]\(../usage/p2p.html#all-to-one-gather\)](#)
 - * [\[All-to-all\]\(../usage/p2p.html#all-to-all\)](#)
 - * [\[Neighbor exchange\]\(../usage/p2p.html#neighbor-exchange\)](#)
- * [\[Thread Safety\]\(../usage/threadsafety.html\)](#)
- * [\[In-place Operations\]\(../usage/inplace.html\)](#)
- * [\[Using NCCL with CUDA Graphs\]\(../usage/cudagraph.html\)](#)
- * [\[User Buffer Registration\]\(../usage/bufferreg.html\)](#)
 - * [\[NVLink Sharp Buffer Registration\]\(../usage/bufferreg.html#nvlink-sharp-buffer-registration\)](#)
 - * [\[IB Sharp Buffer Registration\]\(../usage/bufferreg.html#ib-sharp-buffer-registration\)](#)
 - * [\[General Buffer Registration\]\(../usage/bufferreg.html#general-buffer-registration\)](#)
 - * [\[Memory Allocator\]\(../usage/bufferreg.html#memory-allocator\)](#)
- * [\[NCCL API\]\(../api.html\)](#)

* Communicator Creation and Management Functions

- * `ncclGetLastError`
- * `ncclGetErrorString`
- * `ncclGetVersion`
- * `ncclGetUniqueId`
- * `ncclCommInitRank`
- * `ncclCommInitAll`
- * `ncclCommInitRankConfig`
- * `ncclCommInitRankScalable`
- * `ncclCommSplit`
- * `ncclCommFinalize`
- * `ncclCommDestroy`
- * `ncclCommAbort`
- * `ncclCommGetAsyncError`
- * `ncclCommCount`
- * `ncclCommCuDevice`
- * `ncclCommUserRank`
- * `ncclCommRegister`
- * `ncclCommDeregister`
- * `ncclMemAlloc`
- * `ncclMemFree`

* [Collective Communication Functions](colls.html)

- * `[ncclAllReduce](colls.html#ncclallreduce)`
- * `[ncclBroadcast](colls.html#ncclbroadcast)`
- * `[ncclReduce](colls.html#ncclreduce)`
- * `[ncclAllGather](colls.html#ncclallgather)`
- * `[ncclReduceScatter](colls.html#ncclreducescatter)`

- * [Group Calls](group.html)
- * [ncclGroupStart](group.html#ncclgroupstart)
- * [ncclGroupEnd](group.html#ncclgroupend)
- * [ncclGroupSimulateEnd](group.html#ncclgroupsimulateend)
- * [Point To Point Communication Functions](p2p.html)
- * [ncclSend](p2p.html#ncclsend)
- * [ncclRecv](p2p.html#ncclrecv)
- * [Types](types.html)
- * [ncclComm_t](types.html#ncclcomm-t)
- * [ncclResult_t](types.html#ncclresult-t)
- * [ncclDataType_t](types.html#nccldatatype-t)
- * [ncclRedOp_t](types.html#ncclredop-t)
- * [ncclScalarResidence_t](types.html#ncclscalarresidence-t)
- * [ncclConfig_t](types.html#ncclconfig-t)
- * [ncclSimInfo_t](types.html#ncclsiminfo-t)
- * [User Defined Reduction Operators](ops.html)
- * [ncclRedOpCreatePreMulSum](ops.html#ncclredopcreatepremulsum)
- * [ncclRedOpDestroy](ops.html#ncclredopdestroy)
- * [Migrating from NCCL 1 to NCCL 2](../nccl1.html)
- * [Initialization](../nccl1.html#initialization)
- * [Communication](../nccl1.html#communication)
- * [Counts](../nccl1.html#counts)
- * [In-place usage for AllGather and ReduceScatter](../nccl1.html#in-place-usage-for-allgather-and-reducescatter)
- * [AllGather arguments order](../nccl1.html#allgather-arguments-order)
- * [Datatypes](../nccl1.html#datatypes)
- * [Error codes](../nccl1.html#error-codes)

- * [Examples](../examples.html)
 - * [Communicator Creation and Destruction Examples](../examples.html#communicator-creation-and-destruction-examples)
 - * [Example 1: Single Process, Single Thread, Multiple Devices](../examples.html#example-1-single-process-single-thread-multiple-devices)
 - * [Example 2: One Device per Process or Thread](../examples.html#example-2-one-device-per-process-or-thread)
 - * [Example 3: Multiple Devices per Thread](../examples.html#example-3-multiple-devices-per-thread)
 - * [Example 4: Multiple communicators per device](../examples.html#example-4-multiple-communicators-per-device)
 - * [Communication Examples](../examples.html#communication-examples)
 - * [Example 1: One Device per Process or Thread](../examples.html#example-1-one-device-per-process-or-thread)
 - * [Example 2: Multiple Devices per Thread](../examples.html#example-2-multiple-devices-per-thread)
 - * [NCCL and MPI](../mpi.html)
 - * [API](../mpi.html#api)
 - * [Using multiple devices per process](../mpi.html#using-multiple-devices-per-process)
 - * [ReduceScatter operation](../mpi.html#reducescatter-operation)
 - * [Send and Receive counts](../mpi.html#send-and-receive-counts)
 - * [Other collectives and point-to-point operations](../mpi.html#other-collectives-and-point-to-point-operations)
 - * [In-place operations](../mpi.html#in-place-operations)
 - * [Using NCCL within an MPI Program](../mpi.html#using-nccl-within-an-mpi-program)
 - * [MPI Progress](../mpi.html#mpi-progress)
 - * [Inter-GPU Communication with CUDA-aware

[MPI\]\(../mpi.html#inter-gpu-communication-with-cuda-aware-mpi\)](#)

* [\[Environment Variables\]\(../env.html\)](#)

* [\[System configuration\]\(../env.html#system-configuration\)](#)

* [\[NCCL_SOCKET_IFNAME\]\(../env.html#nccl-socket-ifname\)](#)

* [\[Values accepted\]\(../env.html#values-accepted\)](#)

* [\[NCCL_SOCKET_FAMILY\]\(../env.html#nccl-socket-family\)](#)

* [\[Values accepted\]\(../env.html#id2\)](#)

* [\[NCCL_SOCKET_RETRY_CNT\]\(../env.html#nccl-socket-retry-cnt\)](#)

* [\[Values accepted\]\(../env.html#id3\)](#)

* [\[NCCL_SOCKET_RETRY_SLEEP_MSEC\]\(../env.html#nccl-socket-retry-sleep-msec\)](#)

* [\[Values accepted\]\(../env.html#id4\)](#)

* [\[NCCL_SOCKET_NTHREADS\]\(../env.html#nccl-socket-nthreads\)](#)

* [\[Values accepted\]\(../env.html#id5\)](#)

* [\[NCCL_NSOCKS_PERTHREAD\]\(../env.html#nccl-nsocks-perthread\)](#)

* [\[Values accepted\]\(../env.html#id6\)](#)

* [\[NCCL_CROSS_NIC\]\(../env.html#nccl-cross-nic\)](#)

* [\[Values accepted\]\(../env.html#id7\)](#)

* [\[NCCL_IB_HCA\]\(../env.html#nccl-ib-hca\)](#)

* [\[Values accepted\]\(../env.html#id8\)](#)

* [\[NCCL_IB_TIMEOUT\]\(../env.html#nccl-ib-timeout\)](#)

* [\[Values accepted\]\(../env.html#id9\)](#)

* [\[NCCL_IB_RETRY_CNT\]\(../env.html#nccl-ib-retry-cnt\)](#)

* [\[Values accepted\]\(../env.html#id10\)](#)

* [\[NCCL_IB_GID_INDEX\]\(../env.html#nccl-ib-gid-index\)](#)

* [\[Values accepted\]\(../env.html#id11\)](#)

* [\[NCCL_IB_ADDR_FAMILY\]\(../env.html#nccl-ib-addr-family\)](#)

* [\[Values accepted\]\(../env.html#id12\)](#)

* [NCCL_IB_ADDR_RANGE](../env.html#nccl-ib-addr-range)

* [Values accepted](../env.html#id13)

* [NCCL_IB_ROCE_VERSION_NUM](../env.html#nccl-ib-roce-version-num)

* [Values accepted](../env.html#id14)

* [NCCL_IB_SL](../env.html#nccl-ib-sl)

* [Values accepted](../env.html#id15)

* [NCCL_IB_TC](../env.html#nccl-ib-tc)

* [Values accepted](../env.html#id16)

* [NCCL_IB_FIFO_TC](../env.html#nccl-ib-fifo-tc)

* [Values accepted](../env.html#id17)

* [NCCL_IB_RETURN_ASYNC_EVENTS](../env.html#nccl-ib-return-async-events)

* [Values accepted](../env.html#id18)

* [NCCL_OOB_NET_ENABLE](../env.html#nccl-oob-net-enable)

* [Values accepted](../env.html#id19)

* [NCCL_OOB_NET_IFNAME](../env.html#nccl-oob-net-ifname)

* [Values accepted](../env.html#id20)

* [NCCL_UID_STAGGER_THRESHOLD](../env.html#nccl-uid-stagger-threshold)

* [Values accepted](../env.html#id21)

* [NCCL_UID_STAGGER_RATE](../env.html#nccl-uid-stagger-rate)

* [Values accepted](../env.html#id22)

* [NCCL_NET](../env.html#nccl-net)

* [Values accepted](../env.html#id23)

* [NCCL_NET_PLUGIN](../env.html#nccl-net-plugin)

* [Values accepted](../env.html#id24)

* [NCCL_TUNER_PLUGIN](../env.html#nccl-tuner-plugin)

* [Values accepted](../env.html#id25)

* [NCCL_PROFILER_PLUGIN](../env.html#nccl-profiler-plugin)

- * [Values accepted](../env.html#id26)
- * [NCCL_IGNORE_CPU_AFFINITY](../env.html#nccl-ignore-cpu-affinity)
 - * [Values accepted](../env.html#id27)
- * [NCCL_CONF_FILE](../env.html#nccl-conf-file)
 - * [Values accepted](../env.html#id28)
- * [NCCL_DEBUG](../env.html#nccl-debug)
 - * [Values accepted](../env.html#id30)
- * [NCCL_DEBUG_FILE](../env.html#nccl-debug-file)
 - * [Values accepted](../env.html#id31)
- * [NCCL_DEBUG_SUBSYS](../env.html#nccl-debug-subsys)
 - * [Values accepted](../env.html#id32)
- * [NCCL_COLLNET_ENABLE](../env.html#nccl-collnet-enable)
 - * [Value accepted](../env.html#value-accepted)
- * [NCCL_COLLNET_NODE_THRESHOLD](../env.html#nccl-collnet-node-threshold)
 - * [Value accepted](../env.html#id33)
- * [NCCL_TOPO_FILE](../env.html#nccl-topo-file)
 - * [Value accepted](../env.html#id34)
- * [NCCL_TOPO_DUMP_FILE](../env.html#nccl-topo-dump-file)
 - * [Value accepted](../env.html#id35)
- * [NCCL_SET_THREAD_NAME](../env.html#nccl-set-thread-name)
 - * [Value accepted](../env.html#id36)
- * [Debugging](../env.html#debugging)
- * [NCCL_P2P_DISABLE](../env.html#nccl-p2p-disable)
 - * [Values accepted](../env.html#id37)
- * [NCCL_P2P_LEVEL](../env.html#nccl-p2p-level)
 - * [Values accepted](../env.html#id38)
- * [Integer Values (Legacy)](../env.html#integer-values-legacy)

* [NCCL_P2P_DIRECT_DISABLE](../env.html#nccl-p2p-direct-disable)

* [Values accepted](../env.html#id39)

* [NCCL_SHM_DISABLE](../env.html#nccl-shm-disable)

* [Values accepted](../env.html#id40)

* [NCCL_BUFFSIZE](../env.html#nccl-buffersize)

* [Values accepted](../env.html#id41)

* [NCCL_NTHREADS](../env.html#nccl-nthreads)

* [Values accepted](../env.html#id42)

* [NCCL_MAX_NCHANNELS](../env.html#nccl-max-nchannels)

* [Values accepted](../env.html#id43)

* [NCCL_MIN_NCHANNELS](../env.html#nccl-min-nchannels)

* [Values accepted](../env.html#id44)

* [NCCL_CHECKS_DISABLE](../env.html#nccl-checks-disable)

* [Values accepted](../env.html#id45)

* [NCCL_CHECK_POINTERS](../env.html#nccl-check-pointers)

* [Values accepted](../env.html#id46)

* [NCCL_LAUNCH_MODE](../env.html#nccl-launch-mode)

* [Values accepted](../env.html#id47)

* [NCCL_IB_DISABLE](../env.html#nccl-ib-disable)

* [Values accepted](../env.html#id48)

* [NCCL_IB_AR_THRESHOLD](../env.html#nccl-ib-ar-threshold)

* [Values accepted](../env.html#id49)

* [NCCL_IB_QPS_PER_CONNECTION](../env.html#nccl-ib-qps-per-connection)

* [Values accepted](../env.html#id50)

* [NCCL_IB_SPLIT_DATA_ON_QPS](../env.html#nccl-ib-split-data-on-qps)

* [Values accepted](../env.html#id51)

* [NCCL_IB_CUDA_SUPPORT](../env.html#nccl-ib-cuda-support)

* [Values accepted](../env.html#id52)

* [NCCL_IB_PCI_RELAXED_ORDERING](../env.html#nccl-ib-pci-relaxed-ordering)

* [Values accepted](../env.html#id53)

* [NCCL_IB_ADAPTIVE_ROUTING](../env.html#nccl-ib-adaptive-routing)

* [Values accepted](../env.html#id54)

* [NCCL_IB_ECE_ENABLE](../env.html#nccl-ib-ece-enable)

* [Values accepted](../env.html#id55)

* [NCCL_MEM_SYNC_DOMAIN](../env.html#nccl-mem-sync-domain)

* [Values accepted](../env.html#id56)

* [NCCL_CUMEM_ENABLE](../env.html#nccl-cumem-enable)

* [Values accepted](../env.html#id57)

* [NCCL_CUMEM_HOST_ENABLE](../env.html#nccl-cumem-host-enable)

* [Values accepted](../env.html#id58)

* [NCCL_NET_GDR_LEVEL (formerly

NCCL_IB_GDR_LEVEL)](../env.html#nccl-net-gdr-level-formerly-nccl-ib-gdr-level)

* [Values accepted](../env.html#id59)

* [Integer Values (Legacy)](../env.html#id60)

* [NCCL_NET_GDR_READ](../env.html#nccl-net-gdr-read)

* [Values accepted](../env.html#id61)

* [NCCL_NET_SHARED_BUFFERS](../env.html#nccl-net-shared-buffers)

* [Value accepted](../env.html#id62)

* [NCCL_NET_SHARED_COMMS](../env.html#nccl-net-shared-comms)

* [Value accepted](../env.html#id63)

* [NCCL_SINGLE_RING_THRESHOLD](../env.html#nccl-single-ring-threshold)

* [Values accepted](../env.html#id64)

* [NCCL_LL_THRESHOLD](../env.html#nccl-ll-threshold)

* [Values accepted](../env.html#id65)

* [NCCL_TREE_THRESHOLD](../env.html#nccl-tree-threshold)

* [Values accepted](../env.html#id66)

* [NCCL_ALGO](../env.html#nccl-algo)

* [Values accepted](../env.html#id67)

* [NCCL_PROTO](../env.html#nccl-proto)

* [Values accepted](../env.html#id68)

* [NCCL_NVX_DISABLE](../env.html#nccl-nvx-disable)

* [Value accepted](../env.html#id69)

* [NCCL_P2P_DISABLE](../env.html#nccl-p2p-disable)

* [Value accepted](../env.html#id70)

* [NCCL_P2P_P2P_LEVEL](../env.html#nccl-p2p-p2p-level)

* [Value accepted](../env.html#id71)

* [NCCL_RUNTIME_CONNECT](../env.html#nccl-runtime-connect)

* [Value accepted](../env.html#id72)

* [NCCL_GRAPH_REGISTER](../env.html#nccl-graph-register)

* [Value accepted](../env.html#id74)

* [NCCL_LOCAL_REGISTER](../env.html#nccl-local-register)

* [Value accepted](../env.html#id75)

* [NCCL_LEGACY_CUDA_REGISTER](../env.html#nccl-legacy-cuda-register)

* [Value accepted](../env.html#id76)

* [NCCL_SET_STACK_SIZE](../env.html#nccl-set-stack-size)

* [Value accepted](../env.html#id77)

* [NCCL_GRAPH_MIXING_SUPPORT](../env.html#nccl-graph-mixing-support)

* [Value accepted](../env.html#id79)

* [NCCL_DMABUF_ENABLE](../env.html#nccl-dmabuf-enable)

* [Value accepted](../env.html#id80)

* [NCCL_P2P_NET_CHUNKSIZE](../env.html#nccl-p2p-net-chunksize)

- * [Values accepted](../env.html#id81)
- * [NCCL_P2P_LL_THRESHOLD](../env.html#nccl-p2p-ll-threshold)
- * [Values accepted](../env.html#id82)
- * [NCCL_ALLOC_P2P_NET_LL_BUFFERS](../env.html#nccl-alloc-p2p-net-ll-buffers)
- * [Values accepted](../env.html#id83)
- * [NCCL_COMM_BLOCKING](../env.html#nccl-comm-blocking)
- * [Values accepted](../env.html#id84)
- * [NCCL_CGA_CLUSTER_SIZE](../env.html#nccl-cga-cluster-size)
- * [Values accepted](../env.html#id85)
- * [NCCL_MAX_CTAS](../env.html#nccl-max-ctas)
- * [Values accepted](../env.html#id86)
- * [NCCL_MIN_CTAS](../env.html#nccl-min-ctas)
- * [Values accepted](../env.html#id87)
- * [NCCL_NVLS_ENABLE](../env.html#nccl-nvls-enable)
- * [Values accepted](../env.html#id88)
- * [NCCL_IB_MERGE_NICS](../env.html#nccl-ib-merge-nics)
- * [Values accepted](../env.html#id89)
- * [NCCL_MNNVL_ENABLE](../env.html#nccl-mnnvl-enable)
- * [Values accepted](../env.html#id90)
- * [NCCL_RAS_ENABLE](../env.html#nccl-ras-enable)
- * [Values accepted](../env.html#id91)
- * [NCCL_RAS_ADDR](../env.html#nccl-ras-addr)
- * [Values accepted](../env.html#id92)
- * [NCCL_RAS_TIMEOUT_FACTOR](../env.html#nccl-ras-timeout-factor)
- * [Values accepted](../env.html#id93)
- * [Troubleshooting](../troubleshooting.html)
- * [Errors](../troubleshooting.html#errors)

- * [\[RAS\]\(../troubleshooting.html#ras\)](#)
- * [\[RAS\]\(../troubleshooting/ras.html\)](#)
- * [\[Principle of Operation\]\(../troubleshooting/ras.html#principle-of-operation\)](#)
- * [\[RAS Queries\]\(../troubleshooting/ras.html#ras-queries\)](#)
- * [\[Sample Output\]\(../troubleshooting/ras.html#sample-output\)](#)
- * [\[GPU Direct\]\(../troubleshooting.html#gpu-direct\)](#)
- * [\[GPU-to-GPU communication\]\(../troubleshooting.html#gpu-to-gpu-communication\)](#)
- * [\[GPU-to-NIC communication\]\(../troubleshooting.html#gpu-to-nic-communication\)](#)
- * [\[PCI Access Control Services \(ACS\)\]\(../troubleshooting.html#pci-access-control-services-ac\)](#)
- * [\[Topology detection\]\(../troubleshooting.html#topology-detection\)](#)
- * [\[Shared memory\]\(../troubleshooting.html#shared-memory\)](#)
- * [\[Docker\]\(../troubleshooting.html#docker\)](#)
- * [\[Systemd\]\(../troubleshooting.html#systemd\)](#)
- * [\[Networking issues\]\(../troubleshooting.html#networking-issues\)](#)
- * [\[IP Network Interfaces\]\(../troubleshooting.html#ip-network-interfaces\)](#)
- * [\[IP Ports\]\(../troubleshooting.html#ip-ports\)](#)
- * [\[InfiniBand\]\(../troubleshooting.html#infiniband\)](#)

* [\[RDMA over Converged Ethernet](#)

[\(RoCE\)\]\(../troubleshooting.html#rdma-over-converged-ethernet-roce\)](#)

[__\[NCCL\]\(../index.html\)](#)

* [\[Docs\]\(../index.html\)](#) »

* [\[NCCL API\]\(../api.html\)](#) »

* [Communicator Creation and Management Functions](#)

* [\[View page source\]\(../_sources/api/comms.rst.txt\)](#)

* * *

Communicator Creation and Management Functions¶

The following functions are public APIs exposed by NCCL to create and manage the collective communication operations.

ncclGetLastError¶

```
const char* `ncclGetLastError`([ncclComm_t](types.html#c.ncclComm_t  
"ncclComm_t") _ comm_)¶
```

Returns a human-readable string corresponding to the last error that occurred in NCCL. Note: The error is not cleared by calling this function. Please note that the string returned by `ncclGetLastError` could be unrelated to the current call and can be a result of previously launched asynchronous operations, if any.

ncclGetErrorString¶

```
const char* `ncclGetErrorString`([ncclResult_t](types.html#c.ncclResult_t  
"ncclResult_t") _ result_)¶
```

Returns a human-readable string corresponding to the passed error code.

```
## ncclGetVersion
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclGetVersion` (int* _ version_)
```

The `ncclGetVersion` function returns the version number of the currently linked NCCL library. The NCCL version number is returned in `_version_` and encoded as an integer which includes the ``NCCL_MAJOR``, ``NCCL_MINOR`` and ``NCCL_PATCH`` levels. The version number returned will be the same as the ``NCCL_VERSION_CODE`` defined in `_nccl.h_`. NCCL version numbers can be compared using the supplied macro; ``NCCL_VERSION(MAJOR,MINOR,PATCH)``

```
## ncclGetUniqueId
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclGetUniqueId` (ncclUniqueId* _ uniqueId_)
```

Generates an Id to be used in `ncclCommInitRank`. `ncclGetUniqueId` should be called once when creating a communicator and the Id should be distributed to all ranks in the communicator before calling `ncclCommInitRank`. `_uniqueId_` should point to a `ncclUniqueId` object allocated by the user.

```
## ncclCommInitRank
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclCommInitRank`([ncclComm_t](types.html#c.ncclComm_t "ncclComm_t")*_  
comm_, int _ nranks_, ncclUniqueId _ commId_, int _ rank_)
```

Creates a new communicator (multi thread/process version). `_rank_` must be between 0 and `_nranks_ - 1` and unique within a communicator clique. Each rank is associated to a CUDA device, which has to be set before calling `ncclCommInitRank`. `ncclCommInitRank` implicitly synchronizes with other ranks, hence it must be called by different threads/processes or used within `ncclGroupStart/ncclGroupEnd`.

```
## ncclCommInitAll
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclCommInitAll`([ncclComm_t](types.html#c.ncclComm_t "ncclComm_t")*_  
comms_, int _ ndev_, const int*_ devlist_)
```

Creates a clique of communicators (single process version) in a blocking way.

This is a convenience function to create a single-process communicator clique.

Returns an array of `_ndev_` newly initialized communicators in `_comms_`. `_comms_`

should be pre-allocated with size at least

`ndev* sizeof([`ncclComm_t`](types.html#c.ncclComm_t "ncclComm_t")). _devlist_`
defines the CUDA devices associated with each rank. If `_devlist_` is NULL, the
first `_ndev_` CUDA devices are used, in order.

```
## ncclCommInitRankConfig
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")  
`ncclCommInitRankConfig`([ncclComm_t](types.html#c.ncclComm_t "ncclComm_t")*_  
comm_, int _ nranks_, ncclUniqueId _ commId_, int _ rank_,  
[ncclConfig_t](types.html#c.ncclConfig_t "ncclConfig_t")*_ config_)
```

This function works the same way as `_ncclCommInitRank_` but accepts a
configuration argument of extra attributes for the communicator. If `config` is
passed as NULL, the communicator will have the default behavior, as if
`ncclCommInitRank` was called.

See the [Creating a communicator with
options](../usage/communicators.html#init-rank-config) section for details on
configuration options.

```
## ncclCommInitRankScalable
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")  
`ncclCommInitRankScalable`([ncclComm_t](types.html#c.ncclComm_t
```

```
"ncclComm_t")*_ newcomm_, int _ nranks_, int _ myrank_, int _ nld_,
ncclUniqueld*_ commlds_, [ncclConfig_t](types.html#c.ncclConfig_t
"ncclConfig_t")*_ config_)Â¶
```

This function works the same way as `_ncclCommInitRankConfig` but accepts a list of `ncclUniquelds` instead of a single one. If only one `ncclUniqueld` is passed, the communicator will be initialized as if `ncclCommInitRankConfig` was called. The provided `ncclUniquelds` will all be used to initialize the single communicator given in argument.

See the [Creating a communicator with options](../usage/communicators.html#init-rank-config) section for details on how to create and distribute the list of `ncclUniquelds`.

```
## ncclCommSplitÂ¶
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
`ncclCommSplit`([ncclComm_t](types.html#c.ncclComm_t "ncclComm_t") _ comm_,
int _ color_, int _ key_, [ncclComm_t](types.html#c.ncclComm_t
"ncclComm_t")*_ newcomm_, [ncclConfig_t](types.html#c.ncclConfig_t
"ncclConfig_t")*_ config_)Â¶
```

The `_ncclCommSplit` is a collective function and creates a set of new

communicators from an existing one. Ranks which pass the same `_color_` value will be part of the same group; color must be a non-negative value. If it is passed as `_NCCL_SPLIT_NOCOLOR_`, it means that the rank will not be part of any group, therefore returning NULL as newcomm. The value of key will determine the rank order, and the smaller key means the smaller rank in new communicator. If keys are equal between ranks, then the rank in the original communicator will be used to order ranks. If the new communicator needs to have a special configuration, it can be passed as `_config_`, otherwise setting config to NULL will make the new communicator inherit the original communicator's configuration. When split, there should not be any outstanding NCCL operations on the `_comm_`. Otherwise, it might cause a deadlock.

```
## ncclCommFinalize¶
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclCommFinalize`([ncclComm_t](types.html#c.ncclComm_t "ncclComm_t") _comm_)¶
```

Finalize a communicator object `_comm_`. When the communicator is marked as nonblocking, `_ncclCommFinalize_` is a nonblocking function. Successful return from it will set communicator state as `_ncclInProgress_` and indicates the communicator is under finalization where all uncompleted operations and the network-related resources are being flushed and freed. Once all NCCL operations are complete, the communicator will transition to the `_ncclSuccess_`

state. Users can query that state with `_ncclCommGetAsyncError_`.

```
## ncclCommDestroy
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclCommDestroy`([ncclComm_t](types.html#c.ncclComm_t "ncclComm_t") _  
comm_)
```

Destroy a communicator object `_comm_`. `_ncclCommDestroy_` only frees the local resources that are allocated to the communicator object `_comm_` if `_ncclCommFinalize_` was previously called on the communicator; otherwise, `_ncclCommDestroy_` will call `ncclCommFinalize` internally. If `_ncclCommFinalize_` is called by users, users should guarantee that the state of the communicator becomes `_ncclSuccess_` before calling `_ncclCommDestroy_`. In all cases, the communicator should no longer be accessed after `ncclCommDestroy` returns. It is recommended that users call `_ncclCommFinalize_` and then `_ncclCommDestroy_`. This function is an intra-node collective call, which all ranks on the same node should call to avoid a hang.

```
## ncclCommAbort
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclCommAbort`([ncclComm_t](types.html#c.ncclComm_t "ncclComm_t") _ comm_)
```

`_ncclCommAbort_` frees resources that are allocated to a communicator object `_comm_` and aborts any uncompleted operations before destroying the communicator. All active ranks are required to call this function in order to abort the NCCL communicator successfully. For more use cases, please check [\[Fault Tolerance\]\(../usage/communicators.html#ft\)](#).

```
## ncclCommGetAsyncError
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclCommGetAsyncError`([ncclComm_t](types.html#c.ncclComm_t "ncclComm_t") _  
comm_, [ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")* _  
asyncError_)
```

Queries the progress and potential errors of asynchronous NCCL operations.

Operations which do not require a stream argument (e.g. `ncclCommFinalize`) can be considered complete as soon as the function returns `_ncclSuccess_` ;

operations with a stream argument (e.g. `ncclAllReduce`) will return

`_ncclSuccess_` as soon as the operation is posted on the stream but may also

report errors through `ncclCommGetAsyncError()` until they are completed. If the

return code of any NCCL function is `_ncclInProgress_` , it means the operation

is in the process of being enqueued in the background, and users must query

the states of the communicators until all the states become `_ncclSuccess_`

before calling another NCCL function. Before the states change into

`_ncclSuccess_` , users are not allowed to issue CUDA kernel to the streams

being used by NCCL. If there has been an error on the communicator, user should destroy the communicator with ``ncclCommAbort()``. If an error occurs on the communicator, nothing can be assumed about the completion or correctness of operations enqueued on that communicator.

```
## ncclCommCount
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t") `ncclCommCount` (const  
[ncclComm_t](types.html#c.ncclComm_t "ncclComm_t") _ comm_, int* _ count)
```

Returns in `_count_` the number of ranks in the NCCL communicator `_comm_`.

```
## ncclCommCuDevice
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")  
`ncclCommCuDevice` (const [ncclComm_t](types.html#c.ncclComm_t "ncclComm_t") _  
comm_, int* _ device)
```

Returns in `_device_` the CUDA device associated with the NCCL communicator `_comm_`.

```
## ncclCommUserRank
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclCommUserRank`(const [ncclComm_t](types.html#c.ncclComm_t "ncclComm_t") _  
comm_, int*_ rank_)Â¶
```

Returns in `_rank_` the rank of the caller in the NCCL communicator `_comm_`.

```
## ncclCommRegisterÂ¶
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclCommRegister`(const [ncclComm_t](types.html#c.ncclComm_t "ncclComm_t") _  
comm_, void*_ buff_, size_t _ size_, void** _ handle_)Â¶
```

Registers the buffer `_buff_` with `_size_` under communicator `_comm_` for zero-copy communication; `_handle_` is returned for future deregistration. See `_buff_` and `_size_` requirements and more instructions in [User Buffer Registration](../usage/bufferreg.html#user-buffer-reg).

```
## ncclCommDeregisterÂ¶
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t")
```

```
`ncclCommDeregister`(const [ncclComm_t](types.html#c.ncclComm_t "ncclComm_t")  
_ comm_, void*_ handle_)Â¶
```

Deregister buffer represented by `_handle_` under communicator `_comm_`.

```
## ncclMemAlloc¶
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t") `ncclMemAlloc`(void _  
**ptr_, size_t _ size_)¶
```

Allocate a GPU buffer with `_size_`. Allocated buffer head address will be returned by `_ptr_`, and the actual allocated size can be larger than requested because of the buffer granularity requirements from all types of NCCL optimizations.

```
## ncclMemFree¶
```

```
[ncclResult_t](types.html#c.ncclResult_t "ncclResult_t") `ncclMemFree`(void _  
*ptr_)¶
```

Free memory allocated by `_ncclMemAlloc()_`.

[Next](colls.html "Collective Communication Functions") [
Previous](../api.html "NCCL API")

* * *

(C) Copyright 2020, NVIDIA Corporation

Built with [Sphinx](<http://sphinx-doc.org/>) using a

[theme](https://github.com/rtfd/sphinx_rtd_theme) provided by [Read the

Docs](<https://readthedocs.org>).