



DEEPSTREAM SDK 5.0.1 FOR NVIDIA DGPU AND JETSON

RN-09353-003 | October 22, 2020
Advance Information | Subject to Change

5.0.1 Release Notes



TABLE OF CONTENTS

1.0	ABOUT THIS RELEASE.....	3
1.1	What's New.....	3
1.1.1	DS 5.0.1	3
1.1.2	DS 5.0 (GA)	3
1.2	Contents of this Release	5
1.3	Documentation in this Release	5
1.4	Differences with Deepstream 4.0	5
2.0	LIMITATIONS.....	6
3.0	NOTES.....	8
3.1	Applications May Be Deployed in a Docker Container.....	8
3.2	Sample Applications Malfunction if Docker Environment Cannot Support Display.....	11
3.3	Installing DeepStream on Jetson	11
3.4	Triton Inference Server In Deepstream	12

1.0 ABOUT THIS RELEASE

These release notes are for the NVIDIA® DeepStream SDK for NVIDIA® Tesla®, NVIDIA® Jetson AGX Xavier™, NVIDIA® Jetson Xavier™ NX, NVIDIA® Jetson Nano™, and NVIDIA® Jetson™ TX2 platforms.

1.1 WHAT'S NEW

The following new features are supported in this DeepStream SDK release:

1.1.1 DS 5.0.1

Bug fixes in:

- ▶ Demuxer and Dewarper components
- ▶ Python Bindings

1.1.2 DS 5.0 (GA)

- ▶ Support for Triton Inference Server
- ▶ On the fly Model updates
- ▶ Event based Smart Record
- ▶ Cloud to device messaging
- ▶ Improved NVDCF tracker
- ▶ Facility to attach encoded detected Image objects as meta data.
- ▶ Sample application which showcases use of opencv in dsexample plugin
- ▶ Native Red Hat Enterprise Linux support
- ▶ Python Bindings incorporated into SDK
- ▶ Transfer learning toolkit Models from https://github.com/NVIDIA-AI-IOT/deepstream_tlt_apps integrated into SDK
- ▶ Python binding enhancements

- Access to frame image data as NumPy array
- Access to inference output tensor data
- Additional sample applications
 - Probe for image data, then use OpenCV to annotate and save frames to file
 - Probe for inference output tensors to parse in Python
 - USB camera input
 - RTSP stream output
- ▶ Better time-stamp handling for live RTSP cameras
- ▶ DRC stream support for Jetson
- ▶ 10 Bit H264 and H265 stream support
- ▶ Misc. bug fixes and Improved stability
- ▶ Support for flowing Metadata attached before Gst-nv4l2 decoder
- ▶ Gst-nvinfer plugin:
 - Support for TensorRT 7.0+:
 - Explicit Full Dimension Network Support
 - Non-maximum Suppression (NMS) for bounding box Clustering
 - On-the-fly model update (Engine/Plan file only)
 - Support for yolov3-spp detector
 - Support for Mask-RCNN Instance segmentation
- ▶ New plugins:
 - Gst-nvdsanalytics plugin for ROI detection, line crossing and direction detection.
 - Gst-nvinferserver plugin for supporting Triton inference server using C++ client APIs (<https://docs.nvidia.com/deeplearning/sdk/triton-inference-server-master-branch-guide/docs/index.html>).
- ▶ New sample applications:
 - Analytics example: Demonstrates batched analytics like ROI filtering, Line crossing, direction detection and overcrowding.
 - OpenCV example: Demonstrates the use of OpenCV in dsexample plugin.
 - Image as Metadata example: Demonstrates how to attach encoded object image as meta data and save the images in jpeg format.
 - Appsrc and Appsink example: Demonstrates AppSrc and AppSink usage for consuming and giving data from non-DS code respectively.
 - Transfer learning example:

Demonstrates a mechanism to save the images for objects which have lesser confidence. This can further be used for Model training.
 - Mask-RCNN example: Demonstrates Instance segmentation using Mask-RCNN model along with sending mask information to cloud.
- ▶ Source code release for the below plugin:

- Gst-nvdsosd: GStreamer Gst-nvdsosd plugin for overlaying bounding boxes, lines, text, object masks.

Note: Existing DeepStream 4.0 Application can be used with DeepStream 5.0 Refer to the “Application Migration to DeepStream 5.0 from DeepStream 4.X” section in the *NVIDIA DeepStream SDK Development Guide 5.0 Release*.

1.2 CONTENTS OF THIS RELEASE

This release includes the following:

- ▶ The DeepStream SDK. Refer to *NVIDIA DeepStream SDK Development Guide 5.0 Release* for a detailed description of the contents of the DeepStream release package. The *Development Guide* also contains other information to help you get started with DeepStream, including information about system software and hardware requirements and external software dependencies that you must install before you use the SDK.
 - For detailed information about GStreamer plugins, metadata usage, see the *DeepStream Plugin Development Guide*.
 - For detailed troubleshooting questions, see the *DeepStream Troubleshooting Guide*.
 - For frequently asked questions, see the *DeepStream FAQ* guide.
- ▶ DeepStream SDK for dGPU and Jetson Software License Agreement (SLA).
- ▶ `LICENSE.txt` contains the license terms of third-party libraries used.

1.3 DOCUMENTATION IN THIS RELEASE

This release contains the following documentation.

- ▶ *NVIDIA DeepStream SDK Development Guide 5.0 Release*
- ▶ *NVIDIA DeepStream SDK API Reference*
- ▶ *NVIDIA DeepStream Python API Reference*
- ▶ *DeepStream Plugin Development Guide*
- ▶ *DeepStream Troubleshooting Guide*
- ▶ *DeepStream FAQ*
- ▶ *DeepStream Legal Information*

1.4 DIFFERENCES WITH DEEPSTREAM 4.0

These are the major differences from DeepStream 4.0:

- ▶ Bounding box coordinates are now in float data type.

2.0 LIMITATIONS

This section provides details about issues discovered during development and QA but not resolved in this release.

- ▶ With V4L2 codecs only MAX 1024 (decode + encode) instances are provided. The maximum number of instances can be increased by doing changes in open source code.
- ▶ `detected-min-w` and `detected-min-h` must be set to values larger than 32 in the primary inference configuration file (`config_infer_primary.txt`) for `gst-dsexample` on Jetson.
- ▶ The Kafka protocol adapter sometimes does not automatically reconnect when the Kafka Broker to which it is connected goes down and comes back up, thereby requiring application restart.
- ▶ If the `nvds` log file, `ds.log`, has been deleted, then to restart logging you must delete the file `/run/rsyslogd.pid` within the container before reenabling logging by running the `setup_nvds_logger.sh` script as described in the “`nvds_logger`” section of the *DeepStream Plugin Development Guide*.
- ▶ On NVIDIA® Jetson AGX Xavier™, more than 50 instances of certain 1080p H.265 streams are not working due to limited memory for the decoder.
- ▶ On Jetson, running a DeepStream application over SSH (via putty) with X11 forwarding does not work.
- ▶ DeepStream currently expects model network width to be a multiple of 4 and network height to be a multiple of 2.
- ▶ Triton Inference Server implementation in DeepStream currently supports a single GPU. The models need to be configured to use a single GPU.
- ▶ For some models sometime output in DeepStream is not exactly same as observed in Transfer Learning Toolkit. This is due to input scaling algorithm differences.
- ▶ Dynamic resolution change support is Alpha quality.
- ▶ On the fly Model update only supports same type of Model with same Network parameters.

- ▶ On Jetson, BT709 input streams are not supported for sample application deepstream-transfer-learning-app and deepstream-image-meta-test.
- ▶ On NVIDIA Tesla T4 with driver 450.51, using NVENC sometimes results into a CUDA error. An alternative is to use software encoder for file and RTSP output.
- ▶ GStreamer sink elements sending upstream QOS events lead to memory leaks in the DeepStream pipeline. Applications should make sure to set the property "qos" to FALSE explicitly. For reference applications, this can be done by setting qos=0 in all sink groups of the configuration file.
- ▶ Multiple networks are simultaneously not supported on a single dla.
- ▶ The sample `objectDetector_FasterRCNN` works only with the default TensorRT plugins that comes as part of TensorRT installation. The samples does not work with opensource plugins library from <https://github.com/NVIDIA/TensorRT>.
- ▶ The YOLOV3, FasterRCNN, SSD, DSSD, RetinaNet and MaskRCNN models from TLT will require a TensorRT open source build. To build TensorRT open source plugins, refer to <https://github.com/NVIDIA/TensorRT>.

3.0 NOTES

- ▶ Optical flow is supported only on dGPUs having Turing architecture and on NVIDIA® Jetson AGX Xavier™ and NVIDIA® Jetson Xavier™ NX.
- ▶ NVIDIA® Jetson Nano™ and NVIDIA® Jetson™ TX2 support only FP16 and FP32 network precisions with NVIDIA® TensorRT™.
- ▶ Jetson AGX Xavier supports INT8, FP16 and FP32 network precisions with TensorRT.
- ▶ On NVIDIA Tesla T4 with driver 450.51, for maximum decoder performance, set the environment variable `INDIVIDUAL_DECODER_SCHEDULING` to `true` before running the application:

```
$export INDIVIDUAL_DECODER_SCHEDULING=TRUE
```

- ▶ The GitHub Models link mentioned in `/opt/nvidia/deepstream/deepstream-5.0/samples/configs/tlt_pretrained_models/README` file is now unavailable. These models are now available here: <https://nvidia.box.com/v/ds-tlt-models>. To learn more about how these models work with DeepStream, refer to https://github.com/NVIDIA-AI-IOT/deepstream_tlt_apps.

3.1 APPLICATIONS MAY BE DEPLOYED IN A DOCKER CONTAINER

Applications built with DeepStream can be deployed using a Docker container, available on NGC (<https://ngc.nvidia.com/>). Sign up for an NVIDIA GPU Cloud account and look for DeepStream containers to get started.

After you sign into your NGC account, navigate to `Dashboard` → `Setup` → `Get API key` to get your `nvcr.io` authentication details.

As an example, you can use the DeepStream 5.0 docker containers on NGC and run the `deepstream-test4-app` sample application as an Azure edge runtime module on your edge device.

The following procedure deploys `deepstream-test4-app`:

- ▶ Using a sample video stream (`sample_720p.h264`)
- ▶ Sending messages with minimal schema
- ▶ Running with display disabled
- ▶ Using message topic `mytopic` (message topic may not be empty)

Set up and install Azure IoT Edge on your system with the instructions provided in the Azure module client README file in the `deepstream5.0` package:

```
<deepstream-5.0_package>/sources/libs/azure_protocol_adaptor/module_client/README
```

Note: For the Jetson platform, omit installation of the moby packages. Moby is currently incompatible with NVIDIA Container Runtime.

See the Azure documentation for information about prerequisites for creating an Azure edge device on the Azure portal:

<https://docs.microsoft.com/en-us/azure/iot-edge/how-to-deploy-modules-portal#prerequisites>

To deploy `deepstream-test4-app` as an Azure IoT edge runtime module

1. On the Azure portal, click the IoT edge device you have created and click Set Modules.
2. Enter these settings:

Container Registry Settings:

Name: `NGC`

Address: `nvcr.io`

User name: `$(authn token)`

Password: *use the password or API key from your NGC account*

Deployment modules:

Add a new module with the name `ds`

Image URI:

- For x86 dockers:

```
docker pull nvcr.io/nvidia/deepstream:5.0.1-20.09-devel
docker pull nvcr.io/nvidia/deepstream:5.0.1-20.09-samples
docker pull nvcr.io/nvidia/deepstream:5.0.1-20.09-iot
docker pull nvcr.io/nvidia/deepstream:5.0.1-20.09-base
```

- For Jetson dockers:

```
docker pull nvcr.io/nvidia/deepstream-14t:5.0.1-20.09-samples
docker pull nvcr.io/nvidia/deepstream-14t:5.0.1-20.09-iot
docker pull nvcr.io/nvidia/deepstream-14t:5.0.1-20.09-base
```

Container Create options:

- For Jetson:

```
{
  "HostConfig": {
    "Runtime": "nvidia"
  },
  "WorkingDir": "/opt/nvidia/deepstream/deepstream-5.0/sources/apps/sample_apps/deepstream-test4",
  "ENTRYPOINT": [
    "/opt/nvidia/deepstream/deepstream-5.0/bin/deepstream-test4-app",
    "-i", "/opt/nvidia/deepstream/deepstream-5.0/samples/streams/sample_720p.h264",
    "-p",
    "/opt/nvidia/deepstream/deepstream-5.0/lib/libnvs_azure_edge_proto.so",
    "--no-display",
    "-s",
    "1",
    "--topic",
    "mytopic"
  ]
}
```

- For X86:

```
{
  "HostConfig": {
    "Runtime": "nvidia"
  },
  "WorkingDir": "/opt/nvidia/deepstream/deepstream-5.0/sources/apps/sample_apps/deepstream-test4",
  "ENTRYPOINT": [
    "/opt/nvidia/deepstream/deepstream-5.0/bin/deepstream-test4-app",
    "-i", "/opt/nvidia/deepstream/deepstream-5.0/samples/streams/sample_720p.h264",
    "-p",
    "/opt/nvidia/deepstream/deepstream-5.0/lib/libnvs_azure_edge_proto.so",
    "--no-display",
    "-s",
    "1",
    "--topic",
    "mytopic"
  ]
}
```

3. Specify route options for the module:

- Option 1: Use a default route where every message from every module is sent upstream.

```
{
  "routes": {
    "route": "FROM /messages/* INTO $upstream"
  }
}
```

- Option 2: Specific routes where messages sent upstream can be filtered based on topic name. For example, in the sample test programs, topic name `mytopic` is used for the module name `ds5`:

```
{
  "routes": {
    "route": "FROM /messages/modules/ds5/outputs/mytopic INTO $upstream"
  }
}
```

3.2 SAMPLE APPLICATIONS MALFUNCTION IF DOCKER ENVIRONMENT CANNOT SUPPORT DISPLAY

If the Docker environment cannot support display, the sample applications `deepstream-test1`, `deepstream-test2`, `deepstream-test3`, and `deepstream-test4` do not work as expected.

To correct this problem, you must recompile the test applications after replacing `nveglglessink` with `fakesink`. With `deepstream-test4`, you also have the option to specify `fakesink` by adding the `--no-display` command line switch.

3.3 INSTALLING DEEPSTREAM ON JETSON

1. Download the NVIDIA SDK Manager to install JetPack 4.4 GA and DeepStream SDK.
2. Select all the JetPack 4.4 components and DeepStreamSDK from the "Additional SDKs" section.

Refer to the "Quick Start Guide" section in *NVIDIA DeepStream Development Guide 5.0 Release* for installation of updated NVIDIA V4L2 Gstreamer plugin.

Note:

- **NVIDIA Container Runtime" package shall be installed using JetPack 4.4 GA and is a pre-requisite for all DeepStream L4T docker containers.**

- It is recommended to use [SD card images](#) for Jetson Nano instead of flashing through SDK Manager. The minimum recommended size for SD cards is 32 GB

3.4 TRITON INFERENCE SERVER IN DEEPSTREAM

Triton inference server on dGPU is supported only via docker container `deepstream:5.0.1-20.09-triton` for x86.

Refer to the *DeepStream Plugin Development Guide* for more details about Triton inference server.

Triton Inference Server Supports following frameworks:

Framework	Tesla	Jetson	Notes / Limitations
TensorRT	Yes	Yes	Supports TensorRT plan or engine file (.plan)
TensorFlow	Yes	Yes	Supports TensorRT optimization Supported model formats: <i>GraphDef</i> or <i>SavedModel</i> Other TF formats such as checkpoint variables or estimators not directly supported
ONNX	Yes	No	Supports TensorRT optimization
PyTorch	Yes	No	PyTorch model must be traced with an example input and saved as a TorchScript Module (.pt)
Caffe2	No	No	Caffe2 Netdef models not supported with DeepStream - Triton

For more information refer to the following links:

- ▶ Triton Inference Server Model Repository:
https://docs.nvidia.com/deeplearning/sdk/triton-inference-server-guide/docs/model_repository.html
Also contains more information on the supported frameworks.
- ▶ TensorRT Optimization in Triton Inference Server for ONNX and Tensorflow:
<https://docs.nvidia.com/deeplearning/sdk/triton-inference-server-guide/docs/optimization.html#framework-specific-optimization>
- ▶ TensorFlow with TensorRT:
<https://docs.nvidia.com/deeplearning/frameworks/tf-trt-user-guide/index.html>
- ▶ Tensorflow Saved Model:
https://www.tensorflow.org/guide/saved_model#the_savedmodel_format_on_disk

Notice

THE INFORMATION IN THIS DOCUMENT AND ALL OTHER INFORMATION CONTAINED IN NVIDIA DOCUMENTATION REFERENCED IN THIS DOCUMENT IS PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE INFORMATION FOR THE PRODUCT, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the product described in this document shall be limited in accordance with the NVIDIA terms and conditions of sale for the product. THE NVIDIA PRODUCT DESCRIBED IN THIS DOCUMENT IS NOT FAULT TOLERANT AND IS NOT DESIGNED, MANUFACTURED OR INTENDED FOR USE IN CONNECTION WITH THE DESIGN, CONSTRUCTION, MAINTENANCE, AND/OR OPERATION OF ANY SYSTEM WHERE THE USE OR A FAILURE OF SUCH SYSTEM COULD RESULT IN A SITUATION THAT THREATENS THE SAFETY OF HUMAN LIFE OR SEVERE PHYSICAL HARM OR PROPERTY DAMAGE (INCLUDING, FOR EXAMPLE, USE IN CONNECTION WITH ANY NUCLEAR, AVIONICS, LIFE SUPPORT OR OTHER LIFE CRITICAL APPLICATION). NVIDIA EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR SUCH HIGH RISK USES. NVIDIA SHALL NOT BE LIABLE TO CUSTOMER OR ANY THIRD PARTY, IN WHOLE OR IN PART, FOR ANY CLAIMS OR DAMAGES ARISING FROM SUCH HIGH RISK USES.

NVIDIA makes no representation or warranty that the product described in this document will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do 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 does not accept any 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.

Other than the right for customer to use the information in this document with the product, no other license, either expressed or implied, is hereby granted by NVIDIA under this document. Reproduction of information in this document is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.

Trademarks

NVIDIA, the NVIDIA logo, TensorRT, Jetson Nano, Jetson AGX Xavier, Jetson Xavier NX, and NVIDIA Tesla are trademarks and/or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright © 2020 NVIDIA Corporation. All rights reserved.