DEEPSTREAM SDK 6.2 FOR NVIDIA DGPU/X86 (NVAIE)

RN-09353-003 | December 15, 2022
Advance Information | Subject to Change

6.2 Release Notes

TABLE OF CONTENTS

1.0 ABOUT THIS RELEASE

These release notes are for the NVIDIA® DeepStream SDK for NVIDIA® Tesla®, NVIDIA® Ampere®, NVIDIA® Hopper®.

1.1 WHAT'S NEW

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

1.1.1 DS 6.2

- Supports Triton 22.09 and Rivermax v1.11.5. Current release is part of NVAIE (https://www.nvidia.com/en-us/data-center/products/ai-enterprise/) and supports only x86/dGPU platforms. Release is through Triton docker: `nvcr.io/nvaie/deepstream-3-0:6.2.0-triton`.
- DeepSORT tracker support.
- REST API support to control DeepStream pipeline on-the-fly (Alpha).
- LIDAR support (Alpha).
- Enable Pre-Processing plugin with SGIE.
- Dewarper enhancements to support additional projections.
- Support Google protobuf encoding and decoding message to message brokers (only Kafka).
- `Nvdsxfer` plugin implementation (NVLink based).
- Enhancements in new Gst-nvstreammux plugin. New `nvstreammux` can be enabled by exporting `USE_NEW_NVSTREAMMUX=yes`. For more information, see the "Gst-nvstreammux" section in the *NVIDIA DeepStream SDK Developer Guide 6.2 Release*.
- Performance optimizations.
- Improved `NVDCF` tracker.
- GPU based drawing for text, line, circles, arrow using `osd` plugin (alpha).

- NVIDIA TAO toolkit (previously called NVIDIA Transfer Learning Toolkit) Models from https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps (branch: `release/tao3.0_ds6.2ga`) integrated into SDK.
- Continued Support for 2D body pose estimation, facial landmark estimation, Emotion recognition, Gaze, Heart Rate, and Gesture. (https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps branch: release/tao3.0_ds6.2ga).
- `nvdsudpsink` plugin optimizations for supporting Mellanox NIC for transmission.
- Improved stability.
- `deepstream-dewarper-app` now supports following new projections: Fisheye to Perspective, Fisheye to Fisheye, Fisheye to Cylindrical, Fisheye to Equirectangular, Fisheye to Panini, Perspective to Equirectangular, Perspective to Panini, Equirectangular to Cylindrical, Equirectangular to Equirectangular, Equirectangular to Fisheye, Equirectangular to Panini, Equirectangular to Perspective, Equirectangular to PushBroom, Equirectangular to Stereographic, Equirectangular to Vertical Radial Cylindrical.
- New plugins/bins:
  - `Gst-nvdsxfer` plugin transfers data over nvlink across multiple GPU under single process.
  - `gst-nvmultiurisrcbin` : The bin to integrate `nvurisrcbin`, and `nvstreammux` into a single `GstBin`. The bin can be configured to act as REST API server.
- New sample applications:
  - DeepStream Server Application: Demonstrates REST API support to control DeepStream pipeline on-the-fly.
  - DeepStream Lidar Inference App: Demonstrates how to setup lidar data reader, lidar data triton inference and lidar data 3D rendering and file dump pipelines over DS3D interfaces and custom libs of `ds3d::dataloader`, `ds3d::datafilter` and `ds3d::datarender`. See more details in the DeepStream Lidar Inference App section in the *NVIDIA DeepStream SDK Developer Guide 6.2 Release*.
  - DeepStream Can Orientation Sample App: Demonstrates can orientation detection with CV-based VPI template matching algorithm. VPI template matching is implemented with DeepStream video template plugin.
  - Sample app to demonstrate loading CUDA memory from `appsrc` in the pipeline.
- Python bindings and samples updates:
  - New samples:
    1. `Deepstream-segmask`: demonstrating usage of `NvOSD_MaskParams` for segmentation.
    2. `Deepstream-imagedata-multistream-cupy`: demonstrating GPU buffer access for decoded images via `CuPy`.

3. `Deepstream-custom-binding-test`: demonstrating use of custom user metadata. This is a sample for the new custom user metadata bindings guide.

- New guides for adding custom bindings including custom user metadata.

- Opensource:
  - Triton plugin `Gst-nvinferserver`
  - Dewarper plugin `Gst-nvdewarper`

1.2 CONTENTS OF THIS RELEASE

This release includes the following:

- The DeepStream SDK. Refer to *NVIDIA DeepStream SDK Developer Guide 6.2 Release* for a detailed description of the contents of the DeepStream release package. The *Developer 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 Guide" section in the *NVIDIA DeepStream SDK Developer Guide 6.2 Release*.
  - For detailed troubleshooting information and frequently asked questions, see the "DeepStream Troubleshooting and FAQ Guide" section in the *NVIDIA DeepStream SDK Developer Guide 6.2 Release*.
- DeepStream SDK for dGPU 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 Developer Guide 6.2 Release*
- *NVIDIA DeepStream SDK API Reference*
- *NVIDIA DeepStream Python API Reference*

1.4 DIFFERENCES WITH DEEPSTREAM 6.1

`gstreamer1.0-libav`, `libav` and `audioparsers` packages are removed in DeepStream dockers. You may install these packages based on your requirement. While running DeepStream applications inside dockers, you may see the following warnings:

```
WARNING from src_elem: No decoder available for type 'audio/mpeg,
mpegversion=(int)4, framed=(boolean)true, stream-format=(string)raw,
level=(string)2, base-profile=(string)lc, profile=(string)lc,
codec_data=(buffer)119056e500, rate=(int)48000, channels=(int)2'.
```

```
Debug info: gsturidecodebin.c(920): unknown_type_cb ():
```

To avoid such warnings, install `gstreamer1.0-libav` and `gstreamer1.0-plugins-good` inside docker.

Specifically for `deepstream-nmos` , `deepstream-avsync-app` and python based `deepstream-imagedata-multistream` app you would need to install `gstreamer1.0-libav` and `gstreamer1.0-plugins-good`.

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.
- The Kafka protocol adapter sometimes does not automatically reconnect when the Kafka Broker to which it is connected goes down and comes back up. This requires the application to restart.
- If the `nvds` log file `ds.log` has been deleted, 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. This is described in the "nvds_logger: Logging Framework" sub-section in the "Gst-nvmsgbroker" section in the *NVIDIA DeepStream Developer Guide 6.2 Release*.
- 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 output in DeepStream is not exactly same as observed in TAO 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.
- Rivermax SDK is not part of DeepStream. So, the following warning is observed (`gst-plugin-scanner:33257`):

```
GStreamer-WARNING **: 11:38:46.882: Failed to load plugin
'/usr/lib/x86_64-linux-gnu/gstreamer-
```

`1.0/deepstream/libnvdsgst_udp.so': librivermax.so.0: cannot open shared object file: No such file or directory`

You can ignore this warning safely.

- RDMA functionality only supported on x86 and only available on the docker.
- There can be performance drop from TensorRT to Triton for some models (5 to 15%).
- To generate the YOLOV3, YOLOV4 and YOLOV4-tiny model engines, the precision of some layers should be specified as FP32 for TensorRT 8.5.X.X limitations. The solution is updated in https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps.
- NVRM: XID errors seen sometimes when running 150+ streams on Ampere and Hopper.
- NVRM: XID errors seen on some setups with `gst-dsexample` and transfer learning sample apps.
- Sometimes during `deepstream-testsr` app execution, assertion " `GStreamer-CRITICAL **: 12:55:35.006: gst_pad_link_full: assertion 'GST_IS_PAD sinkpad)' failed`" is seen which can be safely ignored.
- `Gst-nvdsasr` plugin and `deepstream-avsync-app` is not supported on Hopper GPU.
- While running `deepstream-image-decode-app` assertion being seen which can be safely ignored.
- ASR and TTS plugins are not supported on NVIDIA Hopper since RIVA container for the same not available.
- `deepstream-server` app is not supported with new `nvstreammux` plugin.
- `gst-nvtracker` plugin utilizes gpu0 even when run on other gpu.
- On T4, in vGPU mode, corrupted output is seen visually because of an issue in `gst-nveglglessink` RGBA display path. As a work around, use `nvvideoconvert` plugin with capabilities set to `NV12` before `nveglglessink`. If using `deepstream-app` use `nvdsosd` plugin in GPU mode.
- TAO point-pillar model works only in FP32 mode.
- REST API support for few components (decoder, preprocessor, `nvinfer` along with stream addition deletion support) with limited configuration options. However, user can extend the functionality with steps mentioned in SDK documentation.
- When running OSD in GPU mode this warning seen: "`[cuOSD Warning] at cuosd.cpp:392 : Cannot find any fonts to match Serif, fallback to DejaVuSansMono.ttf`". This can be safely ignored.
- Cross-Branch driver support is not available on vGPU/NVAIE for DeepStream. Both guest & host driver needs to be at `525.60.13`.
- Graph Composer not supported for this release.
- Critical error (`masked_scan_uint32_peek: assertion '(guint64) offset + size <= reader->size - reader->byte' failed`) observed while running python segmentation application but it can be ignored safely.

3.0 NOTES

- Optical flow is supported only on dGPUs having Turing architecture (onwards)
- NVIDIA® DeepStream SDK 6.2 supports TAO 3.0 models (https://developer.nvidia.com/tao-toolkit). For more details, see https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps.
- On vGPU, only cuda device memory `NVBUF_MEM_CUDA_DEVICE` supported.

Note:

- **OpenCV is deprecated by default. But you can enable OpenCV in plugins such as `nvinfer (nvdsinfer)` and `dsexample (gst-dsexample)` by setting `WITH_OPENCV=1` in the `Makefile` of these components. Refer to the component README for more instructions.**
- **When using docker make sure `libopencv-dev` package is installed inside docker if the Application requires it.**

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. The `nvcr.io/nvaie/deepstream-3-0:6.2.0-triton` container will be part of NVAIE 3.0.

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 6.2. 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 `deepstream-6.2` package:

```
<deepstream-
6.2_package>/sources/libs/azure_protocol_adaptor/module_client/README
```

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: $oauthtoken
  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/nvaie/deepstream-3-0:6.2.0-triton
```

- For X86:

```
{
    "HostConfig": {
        "Runtime": "nvidia"
    },
    "WorkingDir": "/opt/nvidia/deepstream/deepstream-
6.2/sources/apps/sample_apps/deepstream-test4",
    "ENTRYPOINT": [
        "/opt/nvidia/deepstream/deepstream-6.2/bin/deepstream-test4-
app",
        "-i", "/opt/nvidia/deepstream/deepstream-
6.2/samples/streams/sample_720p.h264",
        "-p",
        "/opt/nvidia/deepstream/deepstream-
6.2/lib/libnvds_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 `ds`:

```
    {
        "routes": {
            "route": "FROM /messages/modules/ds/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.

Workaround:

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 TRITON INFERENCE SERVER IN DEEPSTREAM

Triton inference server (version 22.09) on dGPU is supported only via docker for x86.

Refer to the *NVIDIA DeepStream Development Guide 6.2 Release* for more details about Triton inference server.

Triton inference server Supports following frameworks:

Framework	Tesla	Notes / Limitations
TensorRT	Yes	Supports TensorRT plan or engine file (.plan)
TensorFlow	Yes	Supports TensorRT optimization Supported model formats: *GraphDef* or *SavedModel* Other TF formats such as checkpoint variables or estimators not directly supported Supports both Tensorflow 1.x and Tensorflow 2.x. Triton defaults to use Tensorflow 1.x. If users need to run Tensorlfow 2.x models, need to update plugin config with: `infer_config{ backend { trt_is { model_repo{ backend_configs { backend: "tensorflow" setting: "version" value: "2" } } } }`
ONNX	Yes	Supports TensorRT optimization
PyTorch	Yes	PyTorch model must be traced with an example input and saved as a TorchScript Module (.pt)

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, NVIDIA Ampere, NVIDIA Hopper and NVIDIA Tesla are trademarks and/or registered trademarks of NVIDIA Corporation in the Unites States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright © 2022 NVIDIA CORPORATION & AFFILIATES. All rights reserved.