DEEPSTREAM SDK 4.0.2 FOR NVIDIA DGPU AND JETSON

RN-09353-003 | December 20, 2019
Advance Information | Subject to Change

4.0.2 Release Notes

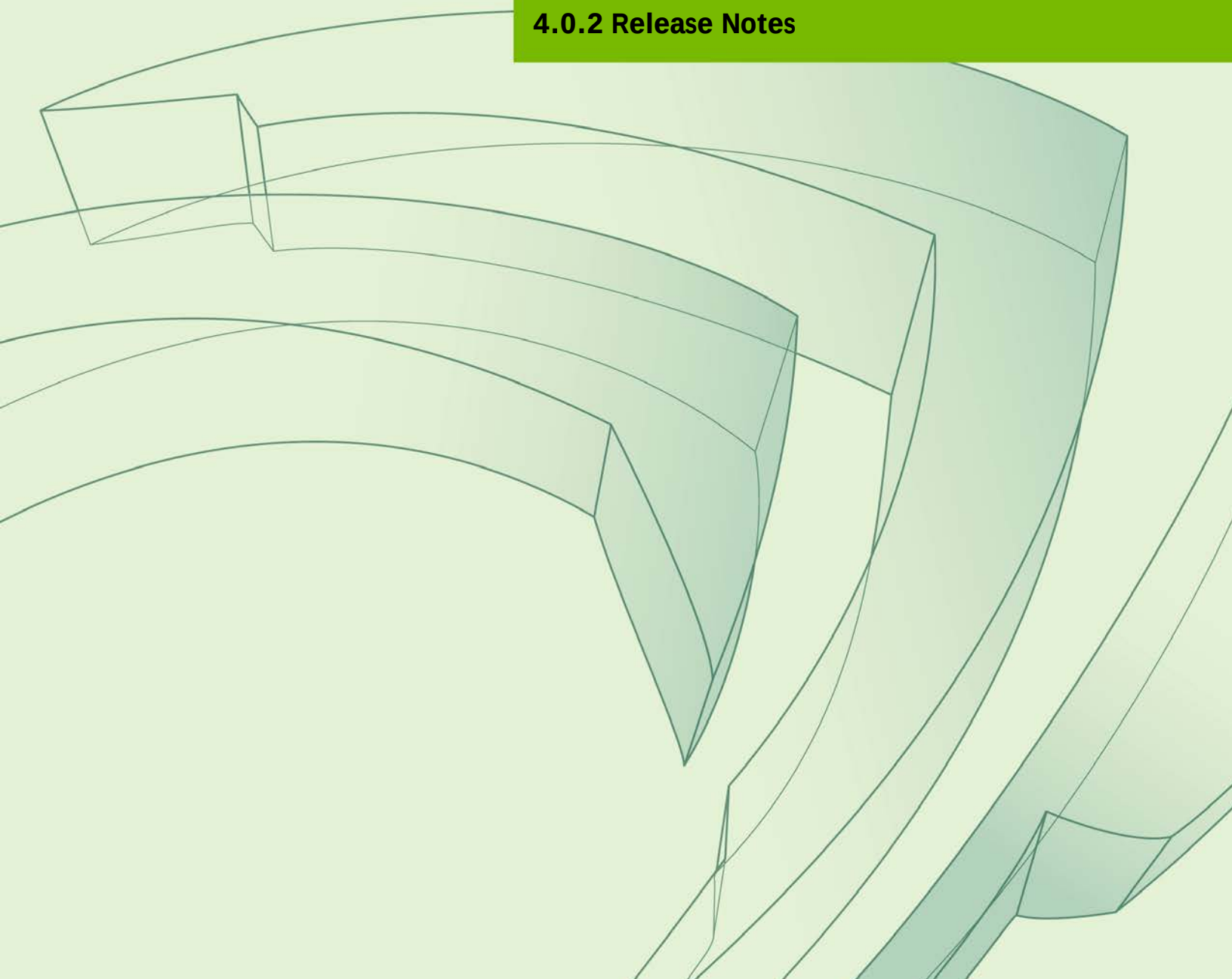

TABLE OF CONTENTS

1.0 ABOUT THIS RELEASE

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

1.1 WHAT'S NEW

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

- Single release for dGPU (Tesla) and Jetson platforms
- New memory management APIs
- New APIs for image conversion and scaling
- New meta data design for ease of use and customization
- Gst-nvinfer plugin enhancements:
  - Encoded model support from Transfer Learning Toolkit SDK. (For information, see https://developer.nvidia.com/transfer-learning-toolkit.)
  - Gray/monochrome model support
  - Support for flowing Tensor output as meta data in pipeline
  - Segmentation model support
  - Configurable support for maintaining aspect ratio when scaling to network resolution
  - New interface for generating an NVIDIA® TensorRT™ engine for custom models which are not UFF/ONNX/Caffe models
  - FP16 and INT8 DLA (Jetson Xavier) support
  - Source code available in DeepStream package
- New plugins:
  - V4L2-based accelerated H265 + H264 encode and decode
  - Accelerated JPEG + MJPEG decoder support
  - Gst-nvvideoconvert for accelerated conversion and scaling

- Gst-nvof, a new Gst plugin for optical flow
- Gst-nvofvisual, a new plugin for visualizing optical flow output
- Gst-nvsegvisual, a new plugin for visualizing segmentation model output

- New sample applications:
  - Yolo: An example Yolo object detector (supporting Yolo v2, v2 tiny, v3, and v3 tiny detectors) showing use of the `IPlugin` interface, custom output parsing, and the CUDA engine generation interface of Gst-nvinfer.
  - Custom meta data example: An example to demonstrate how to add custom/user specific meta data to any component of DeepStream.
  - Dewarper test example: Demonstrates dewarper functionality for single or multiple 360 degree camera streams. Reads camera calibration parameters from a CSV file and renders aisle and spot surfaces on screen.
  - Optical flow test example (for Turing GPU and Jetson Xavier only): Demonstrates optical flow functionality for single or multiple streams. This example uses two GStreamer plugins (Gst-nvof and Gst-nvofvisual). Gst-nvof generates MV (motion vector) data and attaches it as user meta data. Gst-nvofvisual enables visualizing the MV data using a predefined color wheel matrix.
  - Image decode test example: Demonstrates JPEG/MJPEG decode for inferencing.
  - Industrial use case of segmentation: Demonstrates segmentation of multi-stream video or images using a semantic or industrial neural network, and displays output.
  - Example for handling metadata before Gst-nvstreammux: Demonstrates how to set metadata upstream from the Gst-nvstreammux plugin in DeepStream pipeline, and how to access it downstream from that plugin.
  - Gst-nvinfer tensor meta flow example: Demonstrates how to flow and access Gst-nvinfer tensor output as metadata.
  - Performance demo example: Performs single channel cascaded inferencing and object tracking sequentially on all streams in a directory.
  - `deepstream-test5` example: Demonstrates a multistream file/RTSP application using Gst-nvmsgconv and Gst-nvmsgbroker to send metadata to cloud servers. Also shows how to transform a presentation time stamp to a UTC time stamp using RTCP sender reports.

- Miscellaneous features
  - Message broker: Support for Azure IoT.
  - Remote display support for headless server through RTSP out.
  - BT.709 streams supported for dGPU.

- Jetson Docker
- Source Release
  - Gst-nvinfer GStreamer plugin and low level library
  - GStreamer V4L2 codecs plugin

Note: **Due to major feature additions and API changes after DeepStream 3.0, NVIDIA recommends that users of that version migrate their applications and plugins to DeepStream 4.0.2 from scratch.**

1.2 CONTENTS OF THIS RELEASE

This release includes the following:

- The DeepStream SDK. Refer to *NVIDIA DeepStream SDK V4.0.2 Development Guide* 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, Dockers, application and plugin migration to DeepStream 4.0.2, troubleshooting, and a FAQ, see the *DeepStream 4.0.2 Plugin Manual*.

- 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*
- *NVIDIA DeepStream SDK API Reference*
- *NVIDIA DeepStream SDK Plugin Manual*

1.4 DIFFERENCES WITH DEEPSTREAM 3.0

DeepStream 4.0.2 uses a unified code base for Jetson and dGPU platforms. These are the major differences from DeepStream 3.0:

- The 360° camera use case is not supported.
- There are significant modifications to meta data, memory management, conversion/transformation APIs, and the Gst-nvinfer plugin. See *DeepStream 4.0.2 Plugin Manual* for more details.

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.
- Small memory leak observed; fixes in progress.
- `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 Manual*.
- 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.
- Gst-v4l2 decoder does not support Dynamic Resolution Change (DRC) streams due to non-availability of a DRC implementation.
- `Deepstream-test5` is alpha software, and may be expected to evolve in subsequent DeepStream releases.
- The timestamp of the first message from the `test5` application may be out of date. This may happen because the first frame's timestamp is incorrect, as RTCP packets are not arriving immediately to compute the UTC timestamp for a given frame.

- Currently only 8-bit H.264 and H.265 streams are supported.

3.0 NOTES

- Optical flow is only supported on dGPUs having Turing architecture and on NVIDIA® Jetson AGX Xavier™.
- 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.

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 in to your NGC account, go to Dashboard → Setup → Get API key to get your `nvcr.io` authentication details.

As an example, you can use the DeepStream 4.0.2 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 `deepstream4.0.2` package:

```
<deepstream-
4.0.2_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: $oauthtoken
   Password: *use the password or API key from your NGC account*

   Deployment modules:
   Add a new module with the name `ds4`

   Image URI:

   For x86 dockers:
   ```
   docker pull nvcr.io/nvidia/deepstream:4.0.2-19.12-devel
   docker pull nvcr.io/nvidia/deepstream:4.0.2-19.12-samples
   docker pull nvcr.io/nvidia/deepstream:4.0.2-19.12-iot
   docker pull nvcr.io/nvidia/deepstream:4.0.2-19.12-base
   ```

   For Jetson dockers:
   ```
   docker pull nvcr.io/nvidia/deepstream-l4t:4.0.2-19.12-samples
   docker pull nvcr.io/nvidia/deepstream-l4t:4.0.2-19.12-iot
   docker pull nvcr.io/nvidia/deepstream-l4t:4.0.2-19.12-base
   ```

   Container Create options:

   - For Jetson:

   ```
   {
       "HostConfig": {
               "Runtime": "nvidia"
       },
   ```

```
    "WorkingDir":
"/root/deepstream_sdk_v4.0.2_jetson/sources/apps/sample_apps/deepstre
am-test4",
    "ENTRYPOINT": [
        "/opt/nvidia/deepstream/deepstream-4.0/bin/deepstream-test4-
app",
        "-i",
"/root/deepstream_sdk_v4.0.2_jetson/samples/streams/sample_720p.h264"
,
        "-p",
        "/opt/nvidia/deepstream/deepstream-
4.0/lib/libnvds_azure_edge_proto.so",
        "--no-display",
        "-s",
        "1",
        "--topic",
        "mytopic"
    ]
}
```

- For X86:

```
{
    "HostConfig": {
        "Runtime": "nvidia"
    },
    "WorkingDir":
"/root/deepstream_sdk_v4.0.2_x86_64/sources/apps/sample_apps/deepstre
am-test4",
    "ENTRYPOINT": [
        "/opt/nvidia/deepstream/deepstream-4.0/bin/deepstream-test4-
app",
        "-i",
"/root/deepstream_sdk_v4.0.2_x86_64/samples/streams/sample_720p.h264"
,
        "-p",
        "/opt/nvidia/deepstream/deepstream-
4.0/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 `ds4`:

```
    {
        "routes": {
            "route": "FROM /messages/modules/ds4/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 WITH JETSON SOFTWARE DEVELOPER KIT

If you try to install DeepStream alone using a Jetson software developer kit, DeepStream fails, due to DeepStream dependencies on other components.

To install DeepStream with a software developer kit, select all of the JetPack SDK components.

Notice

Trademarks