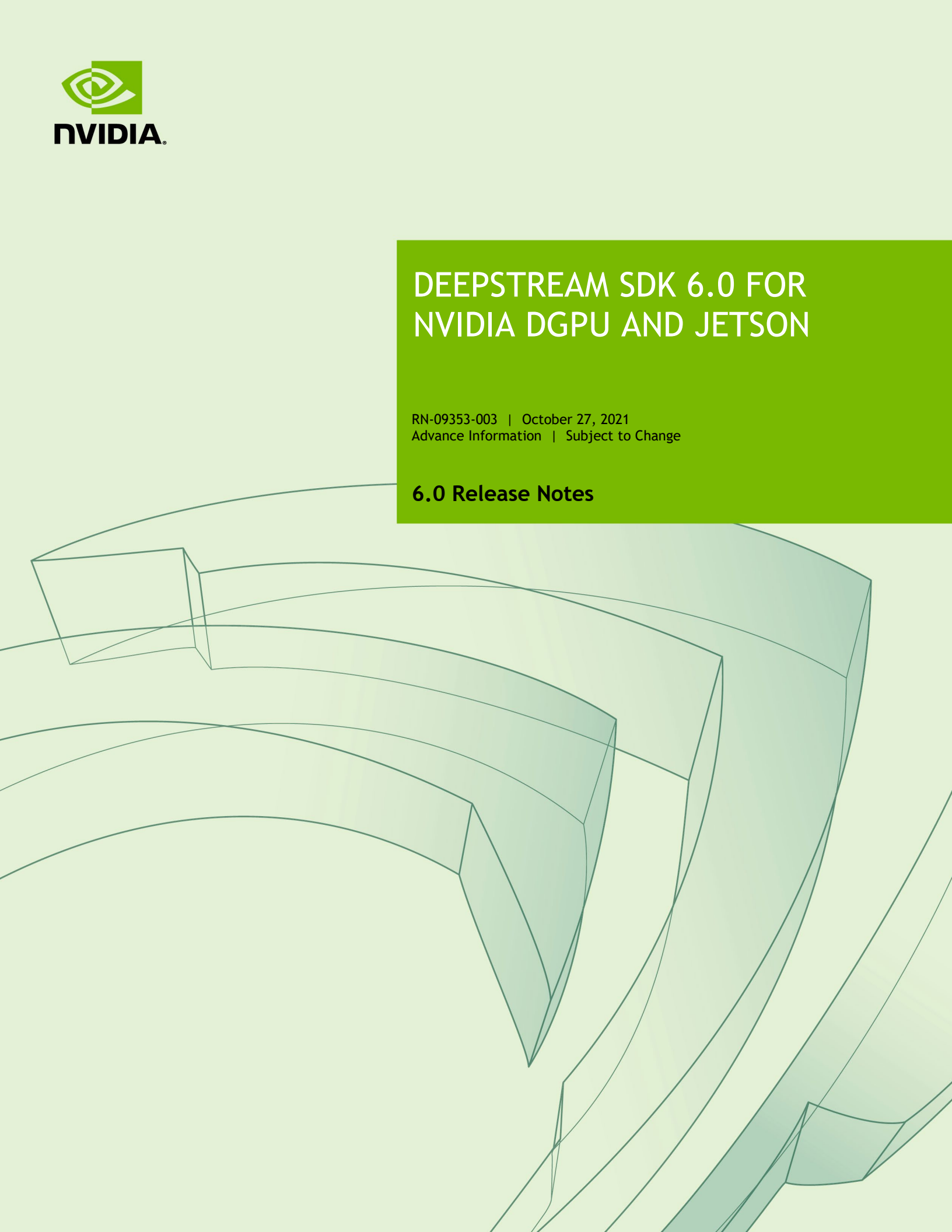

DEEPSTREAM SDK 6.0 FOR NVIDIA DGPU AND JETSON

RN-09353-003 | October 27, 2021
Advance Information | Subject to Change

6.0 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® 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 6.0

- AV sync support (Alpha) for Video Conferencing use cases.
- Mellanox NIC support for receiving Compressed/Uncompressed streams.
- 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_ds6.0ga`).
- Enhancements in new `nvstreammux` (Beta) to support video conferencing use cases.

  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 6.0 Release*.
- Enhancements in Gst-audio/video template plugins.
- Low latency mode support for decoder on dGPU/x86.
- Performance optimizations.
- New DeepSORT tracker (Alpha).
- Improved NVDCF tracker.

Note: **The `KLT tracker` functionality is deprecated. We recommend using `NVDCF tracker`.**

- NVIDIA TAO toolkit (previously called NVIDIA Transfer Learning Toolkit) Models from https://github.com/NVIDIA-AI-IOT/deepstream_tao_apps integrated into SDK.
- Python binding enhancements
  - Bindings source code and build instructions are posted at https://github.com/NVIDIA-AI-IOT/deepstream_python_apps.
  - Cast functions added for all data structs.
- New sample applications in Python:
  - `deepstream-imagedata-multistream-redaction` – multi-stream pipeline with face detection and redaction.
  - `deepstream-rtsp-in-rtsp-out`—multi-stream pipeline with RTSP input/output.
- Gst-nvinfer plugin:
  - NHWC format support
  - TAO converter update for ONNX path
- Gst-nvinferserver plugin:
  - Support for Triton 21.08
  - gRPC support
- New plugins:
  - `gst-nvdspreprocess` plugin demonstrates inference on preprocessed ROIs configured for the streams.
  - `Gst-nvdsasr` plugin for automatic speech recognition (based on NVIDIA Riva SDK).
  - `gst-nvds_text_to_speech` plugin for text to speech conversion (based on NVIDIA Riva SDK).
  - `Gst-nvdsudpsrc` plugin for supporting Mellanox NIC. For more details Refer "Gst-nvdsudpsrc "section in the *NVIDIA DeepStream SDK Developer Guide 6.0 Release.*
- New sample applications:

Note:
- **Refer to README to check prerequisites before running below applications.**

  - Automatic Speech Recognition example: Demonstrates Automatic Speech Recognition functionality.

Note:	**This application can be run inside DeepStream Triton x86 based docker. However, with `gRPC` support in ASR plugin you can run this outside the docker too.**

- AV sync example: Demonstrates synchronization of audio, video, and text output from `nvdsasr` in DeepStream pipeline.

Note:	**This application can be run inside DeepStream Triton x86 based docker. However, with `gRPC` support in ASR plugin you can run this outside the docker too.**

- Action recognition sample: Demonstrates Action recognition using TAO Toolkit 3D and 2D RGB based models.
- Preprocessing plugin sample: Sample application implementation to demonstrate per stream preprocessing on ROIs.
- TTS sample: Demonstrates text to speech conversion functionality along with automatic speech recognition.

DeepStream 5.X Applications can be migrated to DeepStream 6.0. Refer to the "Application Migration to DeepStream 6.0 from DeepStream 5.X" section in the *NVIDIA DeepStream SDK Developer Guide 6.0 Release*.

1.1.2 Graph Composer 1.0.0

- Graph Execution Engine
  - Graph runtime to execute graphs implemented based on Graph Specification
  - Supported on Ubuntu 18.04 x86_64 and NVIDIA Tegra Jetson
- Graph composer tools
  - Composer
    - x86 only (Ubuntu 18.04)
    - User friendly editor view
    - Registry list in view
    - Graph edit with various options
    - Graph open/save
    - Property editor
    - Graph Launcher
    - Container Builder Launcher
    - Registry options
    - Extension Generator
    - Subgraph support
    - Support to execute Graph from Composer UI
    - Extension caching to improve graph install speed
  - Registry CLI

- x86_64 and Jetson (Ubuntu 18.04)
- Local and NVIDIA Cloud repository
- Variants for CUDA 10.2 and 11.4
- Version management based on Semantic versioning
- Command Line Interface tool
- Graph install for graph deploy

- Container Builder CLI

  - x86 only (Ubuntu 18.04)
  - Command Line Interface tool
  - Create container image for graphs
  - Supports .deb/PyPi/.tgz packages

1.2 CONTENTS OF THIS RELEASE

This release includes the following:

- The DeepStream SDK. Refer to *NVIDIA DeepStream SDK Developer Guide 6.0 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 6.0 Release*.
  - For detailed troubleshooting information and frequently asked questions, see the "DeepStream Troubleshooting and FAQ Guide" section in the *NVIDIA DeepStream SDK Developer Guide 6.0 Release*.

- Graph Composer 1.0.0 and DeepStream reference graphs for dGPU and Jetson.
- 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 Developer Guide 6.0 Release*
- *NVIDIA DeepStream SDK API Reference*
- *NVIDIA DeepStream Python API Reference*

1.4 DIFFERENCES WITH DEEPSTREAM 4.0

- Bounding box coordinates are now in float data type.

1.5 DIFFERENCES WITH DEEPSTREAM 5.X

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. Please refer component README for more instructions.

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: Logging Framework" sub-section in the "Gst-nvmsgbroker" section in the *NVIDIA DeepStream Developer Guide 6.0 Release*.
- 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 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.
- `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.

- The sample `objectDetector_FasterRCNN` works only with the default TensorRT plugins that comes as part of TensorRT installation. The samples do not work with opensource plugins library from https://github.com/NVIDIA/TensorRT.
- The YOLOV3, FasterRCNN, SSD, DSSD, RetinaNet and MaskRCNN models from TAO Toolkit will require a TensorRT open-source build. To build TensorRT open-source plugins, refer to https://github.com/NVIDIA/TensorRT.
- Perf regression of 2 to 4 % is seen for T4 platform when used with GPU driver 470.63.01. Work-around is to use previous 465+ driver version.
- DLA numbers on Jetson NX and Jetson AGX are lower compared to DeepStream-5.1 release. This is because, a few layers now run on DLA instead of GPU and those layers are little slow on DLA. This is done to free GPU for other work.
- When running deepstream-app on nano, TX1 and TX2 with default configuration, the following warning is seen: `A lot of buffers are being dropped`. To avoid this warning, use max perf tracker configuration.

  To do so, in `[tracker]` section of the application config file, set `ll-config-file=config_tracker_NvDCF_max_perf.yml`
- Performance of YoloV3, FasterRCNN, SSD, DSSD is lower compared to DeepStream-5.1 release. This is because the new TAO 3.0 models are heavy compared to earlier model version.
- DeepStream cannot be installed on the current 16 GB Xavier NX production modules since Jetpack software takes the entire 16 GB emmc memory space. We recommend using Xavier NX developer kits with 32 GB SD card.
- 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.
- In DeepStream Triton docker `GstVideoAggregator` related warnings might be seen but can be ignored safely.
- Sample graphs containing `NvDsMultiSrcInput` component result in segmentation fault when an error occurs during graph/pipeline initialization.

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.
- NVIDIA® DeepStream SDK 6.0 supports TAO 3.0 models (https://developer.nvidia.com/tao-toolkit). For more details, see https://github.com/NVIDIA-AI-IOT/deepstream_tao_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 6.0 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 `deepstream-6.0` package:

```
<deepstream-
6.0_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: $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/nvidia/deepstream:6.0-devel

     docker pull nvcr.io/nvidia/deepstream:6.0-samples

     docker pull nvcr.io/nvidia/deepstream:6.0-iot

     docker pull nvcr.io/nvidia/deepstream:6.0-base

     docker pull nvcr.io/nvidia/deepstream:6.0-triton
     ```

   - For Jetson dockers:

     ```
     docker pull nvcr.io/nvidia/deepstream-l4t:6.0-samples
     ```

```
docker pull nvcr.io/nvidia/deepstream-l4t:6.0-iot

docker pull nvcr.io/nvidia/deepstream-l4t:6.0-samples

docker pull nvcr.io/nvidia/deepstream-l4t:6.0-triton
```

Container Create options:

- For Jetson:

```
{
    "HostConfig": {
            "Runtime": "nvidia"
    },
    "WorkingDir": "
/opt/nvidia/deepstream/deepstream-
6.0/sources/apps/sample_apps/deepstream-test4",
    "ENTRYPOINT": [
        "/opt/nvidia/deepstream/deepstream-6.0/bin/deepstream-test4-
app",
        "-i", "/opt/nvidia/deepstream/deepstream-6.0/
samples/streams/sample_720p.h264",
        "-p",
"/opt/nvidia/deepstream/deepstream-
6.0/lib/libnvds_azure_edge_proto.so",
        "--no-display",
        "-s",
        "1",
        "--topic",
        "mytopic"
    ]
}
```

- For X86:

```
{
    "HostConfig": {
        "Runtime": "nvidia"
    },
    "WorkingDir": "/opt/nvidia/deepstream/deepstream-
6.0/sources/apps/sample_apps/deepstream-test4",
    "ENTRYPOINT": [
        "/opt/nvidia/deepstream/deepstream-6.0/bin/deepstream-test4-
app",
        "-i", "/opt/nvidia/deepstream/deepstream-
6.0/samples/streams/sample_720p.h264",
        "-p",
        "/opt/nvidia/deepstream/deepstream-
6.0/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 `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 INSTALLING DEEPSTREAM ON JETSON

1. Download the NVIDIA SDK Manager to install JetPack 4.6 GA
2. Select all the JetPack 4.6 components except DeepStreamSDK from the "Additional SDKs" section.

   Refer to the "Quick Start Guide" section in *NVIDIA DeepStream SDK Guide 6.0 Release* for installation of updated NVIDIA BSP packages. Continue with the DeepStream installation instructions after the BSP update.

Note:

- **NVIDIA Container Runtime" package shall be installed using JetPack 4.6 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 `deepstream-l4t:6.0-triton` for x86. On Jetson we support that with or without docker.

Refer to the *NVIDIA DeepStream Development Guide 6.0 Release* 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: *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	Yes	Supports TensorRT optimization
PyTorch	Yes	No	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

Trademarks