

NVIDIA Jetson Customer IQ Migration Tools

User Guide

Version 3.0

Document History

DA_12680-001

Version	Date	Authors	Description of Change
01	December 17, 2025	Naren Bhat	Initial release
02	June 25, 2026	Naren Bhat	Updates for JetPack 7.2
03	August 27, 2026	Naren Bhat; Graham Priestley	Updates for the camera hotfix for JetPack 7.2.1
04	September 23, 2026	Naren Bhat; Graham Priestley	Updated guidance for legacy text-based configuration files and NITO support.

Table of Contents

Contents

Chapter 1. Introduction 5
1.1 Purpose and Scope 5
1.2 Important Terms 5
1.3 System Requirements and Installation 6
1.3.1 Camera Hotfix installation from Windows 7
1.3.2 Camera Hotfix installation from Linux 8
1.3.3 Manually Installing the Camera Hotfix 8
1.3.4 On the host 8
1.3.5 Choosing between the device and host 9
1.4 Support of Text-based ISP Settings Files 9
Chapter 2. Tools 10
2.1 Automatic conversion to NITO 10
2.1.1 Prerequisites 11
2.1.2 The Process in Outline 11
2.1.3 The Process in Detail 12
2.1.4 Environment variables 13
2.2 Manual Conversion to NITO 14
2.2.1 Prerequisites 14
2.2.2 The Process in Outline 15
2.2.3 The Process in Detail 15
2.3 How to Create a Binary CFG File and NITO file 16
2.3.1 camera_overrides.isp 16
2.3.2 Creating a Binary CFG file and NITO file 17
2.4 nvcfg2nito Command Usage 18
2.5 Sample Binary CFG Files 20
2.5.1 nvcfg2nito for Sensor Mode 0 21
2.5.2 nvcfg2nito for Sensor Mode 1 21
2.6 Conclusion 21

List of Tables

Table 1. Terms, Abbreviations, and Acronyms 5
Table 2. nvcfg2nito Command-Line Options 19

Chapter 1. Introduction

1.1 Purpose and Scope

This guide explains how to migrate a text-based configuration file to an NVIDIA Image Tuning Object (NITO) file.

This guide covers nvcfg2nito, a tool that converts binary cfg files containing sensor tuning properties to NITO files. The tool supports only targets that use the NVIDIA® Jetson™ platform with the L4T Argus camera framework.

The package also includes the JetPack 7.2.1 camera hotfix. This guide describes how to install the hotfix on a Jetson device.

1.2 Important Terms

This section defines important acronyms, abbreviations, and terms that are required to understand this document.

Table 1. Terms, Abbreviations, and Acronyms

Term or Abbreviation	Definition
IQ	An acronym for Image Quality. This acronym appears in the names of several NVIDIA products that ensure the quality of processed images.
NITO file	An NVIDIA Image Tuning Object file; a binary file that contains tuning parameters. Image processing software reads tuning parameters from a NITO file. A NITO file can hold tuning parameters for up to eight sensor modes on a single sensor. This is generally enough to accommodate all of the sensor modes utilized in a given system. The NITO file format is supported on NVIDIA DRIVE® and NVIDIA® Jetson™ platforms.
Tuned NITO file	A NITO file whose parameters have been tuned to support a specific set of sensor modes on a specific sensor.

Term or Abbreviation	Definition
Tuning	The process of specifying camera and ISP parameters that are appropriate for a given camera and application.
Tuning library	A library that the NVIDIA Tuning Object Tools use to read, interpret, and save tuning parameters. Tuning libraries are supported on both the DRIVE and Jetson platforms.
Tuning parameters	Parameters that determine the quality of images captured by a camera by configuring the camera and the image processing pipeline.
Binary cfg file	A binary (not human-readable) configuration file that contains tuning parameters for a single sensor and sensor mode. These settings are loaded by the camera driver. Binary cfg files are an older form of configuration file that is supported on the Jetson platform only. nvcfg2nito converts binary cfg files to NITO files, after which the binary cfg files are no longer needed.
Camera Hotfix	This is a tarball or single compressed file that contains a few components that will be needed on the device. It also comes with two installers, one that runs on Ubuntu Linux host and the other on Windows host.

1.3 System Requirements and Installation

Jetson Customer IQ Migration Tools support Windows 11 and Windows 10 hosts.

The package includes a camera hotfix archive for the Jetson device and installers for Windows and Ubuntu hosts. For Ubuntu, version 20.04 LTS is recommended; version 18.04 LTS and later are also supported.

The tool and its supporting files are distributed as a ZIP file. Unpack the ZIP file into your C drive's root folder. It creates a folder named `C:\Jetson_customer_iq_migration_tools`. The following diagram shows that folder's structure.



Figure 1 - Folder Structure

1.3.1 Camera Hotfix installation from Windows

Run the installer from Windows PowerShell in the package's camera_hotfix directory. The camera hotfix archive and installer must be in the same directory.

```
camera_hotfix_installer.ps1 --ip [device_ip_address] --user [user_name]
    --password [user_password]
```

The IP address is required. The username is optional and defaults to “ubuntu.” You can specify the password on the command line. If you omit it, the installer prompts you to enter it securely.

```
PS C:\Jetson_customer_iq_migration_tools> cd camera_hotfix
PS C:\Jetson_customer_iq_migration_tools\camera_hotfix>
.\camera_hotfix_installer.ps1 --ip 192.168.1.50
```

Examples:

```
camera_hotfix_installer.ps1 --ip 192.168.1.50
```

(logs in as 'ubuntu', prompts for that user's password securely)

```
camera_hotfix_installer.ps1 --ip 192.168.1.50 --user myuser
```

(logs in as 'myuser', for customized production images)

```
camera_hotfix_installer.ps1 --ip 192.168.1.50 --password 'my-password'
```

(not recommended)

1.3.2 Camera Hotfix installation from Linux

Run the installer from the package's camera_hotfix directory on an Ubuntu host. The command-line options are the same as those for the Windows installer.

Example:

```
~/temp/installer$ ./camera_hotfix_installer.sh --ip 192.168.1.50
```

1.3.3 Manually Installing the Camera Hotfix

If the camera hotfix installation fails, extract the camera_hotfix tarball and manually copy its contents to the Jetson device. The tarball mirrors the device's directory structure, so copy each file to its corresponding location on the device. For example, copy `/usr/sbin/nvcfg2nito` from the extracted tarball to `/usr/sbin/nvcfg2nito` on the device.

1.3.4 On the host

The package's tuning directory contains `nvcfg2nito.exe`, its Windows tuning library, sample files, and `template.nito`.

The nvcfg2nito executable is located in the folder `tuning\NITO_tools\windows` in the Jetson Customer IQ Migration Tools package. You must add this folder's pathname to the environment variable `PATH`:

```
C:\> set TOOLKIT=C:\Jetson_customer_iq_migration_tools
C:\> set PATH=%TOOLKIT%\tuning\NITO_tools\windows;%PATH%
```

1.3.5 Choosing between the device and host

The camera hotfix installs nvcfg2nito and its tuning library on the Jetson device. Prefer on-device conversion. Use the Windows version in the Jetson Customer IQ Migration Tools package as a fallback if conversion on the device fails.

1.4 Support of Text-based ISP Settings Files

NITO files became the default format for camera ISP settings in JetPack 7.2 GA. The camera hotfix for JetPack 7.2.1 restores support for legacy text-based configuration files. You can continue to use legacy text-based configuration files for ISP tuning; this workflow is highly recommended when using Argus.

If you choose to use NITO files, use the migration tool included in this package to generate them. This hotfix also addresses several issues identified in JetPack 7.2 and JetPack 7.2.1.

Chapter 2. Tools

This chapter describes the process of migrating configuration files to a NITO file and the tools you need to do so.

2.1 Automatic conversion to NITO

This is the preferred method. Set NVCAMERA_NITO_PATH to CONFIG to put the Argus daemon (nvargus-daemon) in legacy text-based configuration mode. Restart the daemon and capture a raw file with nvargus_nvraw. The daemon generates a binary cfg file and automatically converts it to a NITO file.

For a camera with multiple sensor modes, run nvargus_nvraw once for each mode. Each capture updates the same NITO file. The following figure illustrates the workflow.

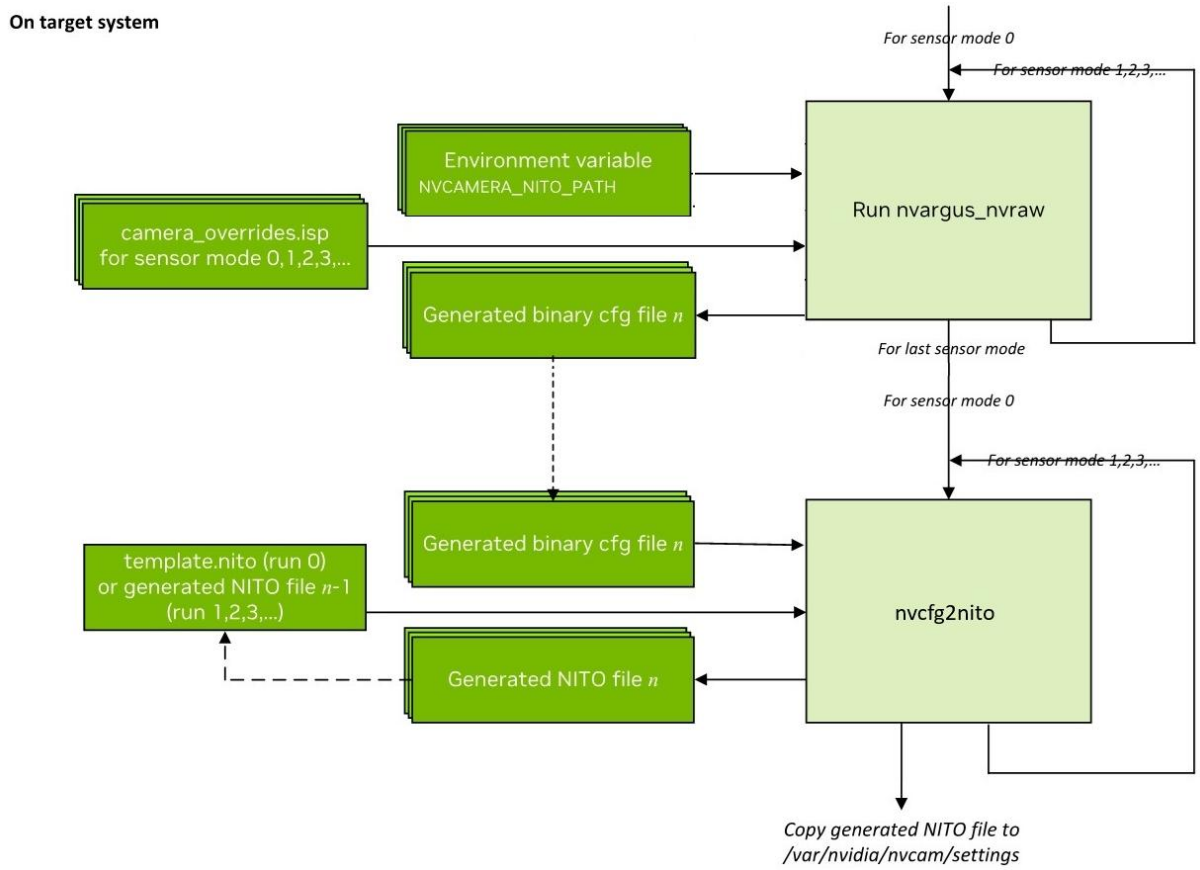


Figure 2 – Automatic conversion workflow

2.1.1 Prerequisites

To perform the migration process, you need:

- A Jetson device that is set up to use a camera with at least one sensor mode.
- The Argus camera framework running on the Jetson device

2.1.2 The Process in Outline

First, run nvargus_nvraw on the Jetson device for a specific sensor mode. The Argus daemon generates a binary cfg file for that mode, converts it, and updates the corresponding mode in the NITO file. Repeat the capture for every sensor mode.

The final NITO file contains the tuning parameters for all captured sensor modes. Copy it to the required location on the Jetson device.

2.1.3 The Process in Detail

This section describes the process outlined in full detail.

1. If you previously attempted NITO conversion on the device, start with a clean output directory. Delete any old NITO file that has the badge name of the camera sensor you are converting. For example, delete /home/ubuntu/camera_badge_name.nito.
2. *On the Jetson device,* for each sensor mode the system uses, from mode 0 to mode $m-1$, create the sensor mode's binary cfg file and also the NITO file. See the next section, How to Create a Binary CFG File and NITO file, for details.
3. After all sensor modes are captured, the NITO file is complete. Its filename is the badge name of the camera module. A successful conversion produces log messages similar to the following:

```
DumpNitoToFile: ---- Finished dumping config data to file:
/home/ubuntu/binary_0.cfg ----
TryAutoConvertBinaryCfgToNito: running: /usr/sbin/nvcfg2nito -t
/usr/lib/aarch64-linux-gnu/nvidia/libnvm_cam_tuning_l4t_cfg2nito.so -s
/var/nvidia/nvcam/settings/template.nito -i /home/ubuntu/binary_0.cfg -o
/home/ubuntu/camera_badge_name.nito -k 0 -v
Updated parameter set:
  ID: UUID(c8278e24-0e47-44e2-9691-36-3b-df-f4-1a-08)
  schema: fd7f5d329b307419b9b76e56cf241b7f83617a4f15f22889050939529070f6d0
  dataSize: 383108
Updated parameters written to Knobset 0 of output file
/home/ubuntu/camera_badge_name.nito
|============================================================|
| The Binary Cfg in use this session was auto-converted to a |
| NITO file.                                                 |
| Binary Cfg file:   /home/ubuntu/binary_0.cfg (sensor mode 0) |
| Auto-converted NITO:  /home/ubuntu/camera_badge_name.nito  |
| If this is a multi-mode sensor, run capture again for each |
| remaining sensor mode; the NITO file above is updated in   |
| place with each additional mode's knobs.                   |
| Whenever you would like to switch this camera module over to |
| NITO, copy the file above to /var/nvidia/nvcam/settings on |
| the Tegra device and clear NVCAMERA_NITO_PATH before       |
| restarting nvargus-daemon.                                 |
| Example: export NVCAMERA_NITO_PATH=                        |
| No action is needed right now: this session continues using |
| the Configuration file as requested.                       |
|============================================================|
```

4. For each new sensor mode, the same NITO file is updated with that mode's knobset. Apart from running nvargus_nvraw for each mode, no additional steps are required.

5. If an error occurs while converting a binary cfg file to NITO, use the manual conversion workflow.
6. To apply the final NITO file, copy it to /var/nvidia/nvcam/settings on the Jetson device and restart the device. Remove the generated NITO file from its original directory, such as /home/ubuntu.

2.1.4 Environment variables

The automatic NITO conversion uses the following environment variables.

2.1.4.1 NVCAMERA_NITO_PATH

By default, NVCAMERA_NITO_PATH is not set and the daemon runs in NITO mode. It searches /var/nvidia/nvcam/settings for a matching NITO file. If a match is found, the Argus daemon loads it and continues. If no match is found, the daemon exits with an error.

Set NVCAMERA_NITO_PATH to CONFIG to use legacy text-based configuration mode. In this mode, the daemon automatically generates the binary cfg and NITO files in the user's home directory, such as /home/ubuntu, and continues using the legacy configuration.

2.1.4.2 NVCAMERA_NITO_DUMP_PATH

The Argus daemon uses NVCAMERA_NITO_DUMP_PATH internally to select the path for the binary cfg file. Normally, do not set it. The following two cases are exceptions.

1. After automatic conversion succeeds, set this environment variable to 0 to prevent the daemon from unnecessarily regenerating or updating the NITO file every time it restarts. This improves startup performance.

   ```
   # export NVCAMERA_NITO_DUMP_PATH=0
   ```

2. You can set a specific binary cfg output path. For example:

   ```
   # export NVCAMERA_NITO_DUMP_PATH=/tmp/binary.cfg
   ```

   This option is not recommended. The Argus daemon normally assigns filenames that account for the sensor mode; do not override that behavior unless necessary.

2.2 Manual Conversion to NITO

This method requires the Jetson device and Windows host. Use this fallback method only if automatic NITO conversion on the device fails.

Converting binary cfg files to a NITO file with nvcfg2nito is a multi-step process, represented in Figure 3.

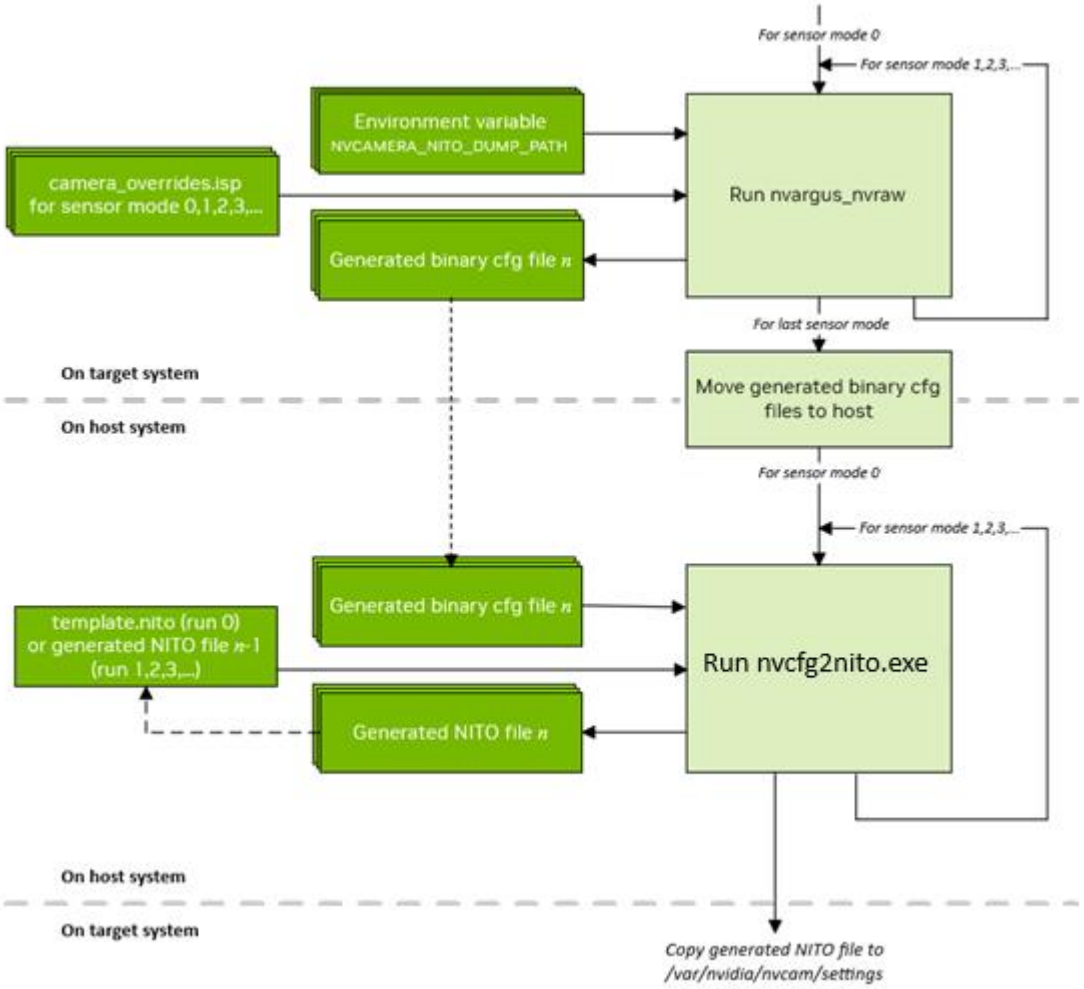


Figure 3. Manual conversion workflow

2.2.1 Prerequisites

To perform the migration process, you need:

- A Jetson device that is programmed to use a camera with at least one sensor mode.
- The Argus camera framework running on the Jetson device

- A Microsoft Windows host computer
- A means of communication between the Jetson device and host computer, such as SCP, that enables you to transfer binary files in both directions

2.2.2 The Process in Outline

First, run nvargus_nvraw on the Jetson device to capture a binary cfg file for a specific sensor mode. Repeat this step for every sensor mode used by the device.

If the device uses m sensor modes, the result is m binary cfg files for modes 0 through m−1.

Second, copy the binary cfg files from the Jetson device to the host system.

Third, on the host, run nvcfg2nito to read binary cfg file 0 and a "template" NITO file, and write a new NITO file that incorporates the settings from binary cfg file 0.

Repeat the conversion for each sensor mode. Each run uses the NITO file generated by the preceding run and adds the parameters for the current mode. Continue until the output contains the parameters for all sensor modes, up to eight.

The final NITO file contains the tuning parameters for all sensor modes. Copy it back to the Jetson device.

2.2.3 The Process in Detail

This section describes the preceding workflow in detail.

1. *On the Jetson device,* for each sensor mode the system uses, from mode 0 to mode m−1, create the sensor mode's binary cfg file. See the next section, How to Create a Binary CFG File, for details.
   a. Copy all of the binary cfg files from the Jetson device to the Windows host where you installed the NITO Tools. Store them in a single folder of your choice.
2. *On the Windows host,* run `nvcfg2nito` for each binary cfg file, 0 through m−1. Specify these command-line options:
   - `-i`: The pathname of this binary cfg file.
   - `-k`: A unique knobSetId for this sensor mode. The knobSetId may be the same as the sensor mode number.
   - `-t`: The name of the tuning library, `nvm_cam_tuning_l4t_cfg2nito.dll`.
   - `-s`: The name of input NITO file.
     - For sensor mode 0, you may use any existing tuned NITO file for this sensor. If you don't have one, use `template.nito`, located in the folder:

```
%TOOLKIT%\tuning\NITO_Tools\TuningFiles\<release>\
```

Where:

- `%TOOLKIT%` is the location of the Migration Tools folder
- `<release>` is the release number of the Migration Tools (as of this writing, `r39.2.1`)

- For each subsequent sensor mode, use the tuned NITO file generated from the preceding mode; that is, the value of the previous run's `-o` parameter.

- `-o`: A filename for the NITO file this run is to generate. Use a logical progression of names for successive sensor modes, such as `tuned_nito_`n`.nito` for modes n=0, 1, 2, ….

For a complete description of the nvcfg2nito command, refer to nvcfg2nito Command Usage.

3. To apply the final generated NITO file to the Jetson device, copy it to the Jetson device's `/var/nvidia/nvcam/settings` directory, and restart the device.

2.3 How to Create a Binary CFG File and NITO file

This section is needed in both cases – Automatic conversion and manual conversion of binary cfg file to NITO.

2.3.1 camera_overrides.isp

A camera overrides file contains tuned parameters that override sensor-specific parameters. The file is optional and can be specific to a sensor mode.

If you use one, copy it to the following directory and set its attributes so that the camera driver can read it.

Filename: camera_overrides.isp
Location on the Jetson device: /var/nvidia/nvcam/settings
File attributes:

- File owner: root:root
- File permission: 644

For example:

```
ubuntu@jetson:~$ sudo su
# chmod 644 /var/nvidia/nvcam/settings/camera_overrides.isp
# chown root:root /var/nvidia/nvcam/settings/camera_overrides.isp
```

Use the camera overrides file as follows:

- Copy the sensor mode specific camera overrides file to the required location, set the file ownership and permissions.
- Follow the steps below of restarting the Argus daemon, and capturing using nvargus_nvraw to create a binary cfg file.
- Repeat the steps for different sensor modes.

2.3.2 Creating a Binary CFG file and NITO file

This procedure applies to both automatic and manual NITO conversion. To create a binary cfg file on a Jetson device, capture an NvRaw file with nvargus_nvraw.

Use this procedure to create a binary cfg file for a sensor mode. During automatic conversion, it also creates or updates the NITO file:

1. Open two terminals on the Jetson device.
2. On terminal 1:

   Optionally, copy the camera overrides file for the sensor mode and set its attributes. Then run the following commands to put the Argus framework in configuration-file mode and restart the Argus daemon. Replace n with the sensor mode number.

```
ubuntu@jetson:~$ sudo su
# pkill nvargus-daemon

# export NVCAMERA_NITO_PATH=CONFIG
# nvargus-daemon
```

3. On terminal 2:

   To capture an NvRaw file (the preferred method for creating a binary cfg file), run nvargus_nvraw. Replace n with the sensor mode number.

```
$ nvargus_nvraw --file /home/ubuntu/sample.nvraw --c sensorId --mode n
```

   sensorId is the zero-based sensor ID when the system has more than one sensor. Typical values are 0, 1, 2, and 3. If you omit --c, the sensor ID defaults to 0.

   nvargus_nvraw creates a binary cfg and NITO file. It also creates an NvRaw file at the path specified by --file, but the NvRaw file is not used in this migration.

Each time you run nvargus_nvraw in terminal 2 for a sensor mode, the following message appears in terminal 1. In this example, binary_0.cfg is for mode 0 and camera_badge_name.nito uses the camera module's badge name.

```
|=============================================================|
| The Binary Cfg in use this session was auto-converted to a  |
| NITO file.                                                  |
| Binary Cfg file:   /home/ubuntu/binary_0.cfg (sensor mode 0) |
| Auto-converted NITO:   /home/ubuntu/camera_badge_name.nito   |
| If this is a multi-mode sensor, run capture again for each  |
| remaining sensor mode; the NITO file above is updated in    |
| place with each additional mode's knobs.                    |
| Whenever you would like to switch this camera module over to |
| NITO, copy the file above to /var/nvidia/nvcam/settings on  |
| the Tegra device and clear NVCAMERA_NITO_PATH before        |
| restarting nvargus-daemon.                                  |
| Example: export NVCAMERA_NITO_PATH=                         |
| No action is needed right now: this session continues using |
| the Configuration file as requested.                        |
|=============================================================|
```

Summary:

- In terminal 1, set NVCAMERA_NITO_PATH to CONFIG and restart nvargus-daemon. No further action is required in that terminal.
- In terminal 2, run nvargus_nvraw for every supported sensor mode, from 0 through m−1.
- When done, you will see the following:
    - For manual conversion on the Windows host, use the uniquely named files such as binary_0.cfg and binary_1.cfg from the user's home directory.
    - For automatic conversion, use the NITO file from the output directory, such as /home/ubuntu/camera_badge_name.nito.

2.4 nvcfg2nito Command Usage

This section describes the nvcfg2nito command for manually converting a binary cfg file to NITO.

The format of the command is:

```
C:\> nvcfg2nito -t <tuninglib> -i <inputcfgbinary> -o <outputnito> ^
                -s <sourcefile>  -k <knobsetid>
```

The following table describes the command-line options.

Table 2. nvcfg2nito Command-Line Options

Elements	Description	Required/Optional
-t <tuninglib or --tuninglibrary <tuninglib>	Specifies a tuning library file (not a folder). nvcfg2nito uses this library to validate the parameter values in its output.	Required
-i <inputcfgbinary> or --inputfile <inputcfgbinary>	Specifies a binary cfg file to be converted to a NITO file.	Required
-s <sourcefile> or --objsource <sourcefile>	Specifies a source NITO file. This may be a NITO file generated by the preceding step in the conversion process, an existing tuned NITO file for the sensor module, or a nonspecific NITO file like `template.nito`. The tool uses it as a source of default values for parameters that are not present in the binary cfg file.	Required
-k <knobsetid> or --knobsetid <knobsetid>	ID of the KnobSet to be updated with the contents of the input binary cfg file. It is typically the sensor mode index. The value's range is [0, 7]. If not specified, it defaults to 0.	Optional
-o <outputnito> or --outputfile <outputnito>	Specifies the pathname of the NITO file that the tool is to generate. This NITO file contains a complete set of parameter values, each one copied from the binary cfg file if it is present there, or the source NITO file if not.	Required

Return code

If nvcfg2nito is successful, it returns an exit code of 0 (zero). If it cannot find and open any of its input files, or if its output validation fails, it returns a non-zero exit code and does not generate output.

Example

This example is intended to illustrate usage of the nvcfg2nito command. A later example, Sample Binary CFG Files, demonstrates the entire process of creating a NITO file.

In this example, two binary cfg files were captured from sensor modes 0 and 1. Run nvcfg2nito once for each file.

The first command must specify:

- The binary cfg file captured for sensor mode 0, with the `-i` option
- A tuned NITO file as the source NITO, with the `-s` option
- KnobSet ID 0 for sensor mode 0, with the `-k` option
- A name for an intermediate generated NITO file, with the `-o` option

The command line looks like this:

```
C:\> nvcfg2nito -t nvm_cam_tuning_l4t_cfg2nito.dll  ^
          -i binary_0.cfg                          ^
          -s tuned_or_template.nito                ^
          -k 0                                     ^
          -o output_0.nito
```

The second command must specify:

- The binary cfg file captured for sensor mode 1, with the `-i` option
- The intermediate NITO file generated by the first command as the source NITO, with the `-s` option
- Knobset ID 1 for sensor mode 1, with the `-k` option
- A name for the final generated NITO file, with the `-o` option

The command line looks like this:

```
C:\> nvcfg2nito -t nvm_cam_tuning_l4t_cfg2nito.dll  ^
          -i binary_1.cfg                          ^
          -s output_0.nito                         ^
          -k 1                                     ^
          -o output_0_1.nito
```

The final generated file, `output_0_1.nito`, has the migrated parameters from both sensor modes. The generated file is ready to be used on the device. It needs only to be copied to the following directory on the device for the driver to pick it up:

```
/var/nvidia/nvcam/settings
```

2.5 Sample Binary CFG Files

The Jetson Customer IQ Migration Tools package includes sample binary cfg files for sensor modes 0 and 1. Use these samples to become familiar with nvcfg2nito when performing manual conversion on a Windows host.

First, set the environment variable `NITO_TOOLS` to the path of the folder that contains the Jetson Customer IQ Migration Tools:

```
C:\> set TOOLKIT=C:\Jetson_customer_iq_migration_tools
C:\> set NITO_TOOLS=%TOOLKIT%\tuning\NITO_tools
```

2.5.1 nvcfg2nito for Sensor Mode 0

Next, run nvcfg2nito with the file `template.nito` as the source file, generating the file `output_0.nito`:

```
nvcfg2nito ^
        -t %NITO_TOOLS%\windows\r39.2.1\nvm_cam_tuning_l4t_cfg2nito.dll   ^
        -s %NITO_TOOLS%\TuningFiles\r39.2.1\template.nito                 ^
        -i %NITO_TOOLS%\Samples\binaryCfg\r39.2.1\sample_binary_0.cfg     ^
        -o %NITO_TOOLS%\TuningFiles\r39.2.1\output_0.nito                 ^
        -k 0
```

2.5.2 nvcfg2nito for Sensor Mode 1

Finally, run nvcfg2nito with the preceding run's generated NITO file as the source file, generating the file `output_0_1.nito`:

```
nvcfg2nito ^
        -t %NITO_TOOLS%\windows\r39.2.1\nvm_cam_tuning_l4t_cfg2nito.dll   ^
        -s %NITO_TOOLS%\TuningFiles\r39.2.1\output_0.nito                 ^
        -i %NITO_TOOLS%\Samples\binaryCfg\r39.2.1\sample_binary_1.cfg     ^
        -o %NITO_TOOLS%\TuningFiles\r39.2.1\output_0_1.nito               ^
        -k 1
```

The generated output_0_1.nito contains the tuning parameters for sensor modes 0 and 1. To apply it, copy it to /var/nvidia/nvcam/settings on the Jetson device.

2.6 Conclusion

- Automatic conversion runs entirely on the Jetson device. After a successful conversion, copy the generated NITO file from the user's home directory to /var/nvidia/nvcam/settings. The generated binary cfg files are no longer needed.

- After automatic conversion succeeds, set NVCAMERA_NITO_DUMP_PATH to 0 as described in the “Environment variables” section. This prevents unnecessary NITO regeneration and improves daemon startup performance.
- If automatic conversion fails, copy the mode-specific binary cfg files, such as binary_0.cfg and binary_1.cfg, to the Windows host and follow the manual conversion procedure.
- Generated files use the camera sensor's module badge name. If two identical sensors have different badge names, each sensor requires a separate NITO file, even when they use identical tuning parameters.
- Process sensor modes in order, starting with mode 0 and continuing through mode m−1.
- NITO generation is cumulative during both automatic and manual conversion.
    - If no NITO file matches the camera badge name, the first conversion starts with template.nito and updates the section for the captured sensor mode.
    - Each subsequent capture uses the preceding NITO output and updates the section for the current sensor mode.
    - For a four-mode sensor, the first run uses template.nito and the next three runs use the output from the preceding run.
    - Before the first run, delete any stale NITO file with the same badge name from the output directory.
- After all automatic-conversion captures are complete, copy the generated NITO file to the /var/nvidia/nvcam/settings directory on the Jetson device.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform 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 accepts no 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.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo, and Jetson are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2026 NVIDIA Corporation. All rights reserved.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com