



Arm[®] Neural Graphics SDK for Game Engines

Version 1.0

Developer guide

Non-Confidential

Copyright © 2025 Arm Limited (or its affiliates).
All rights reserved.

Issue 01

111167_0100_01_en



Arm® Neural Graphics SDK for Game Engines Developer guide

This document is Non-Confidential.

Copyright © 2025 Arm Limited (or its affiliates). All rights reserved.

This document is protected by copyright and other intellectual property rights.

Arm only permits use of this document if you have reviewed and accepted [Arm's Proprietary Notice](#) found at the end of this document.

This document (111167_0100_01_en) was issued on 2025-10-27. There might be a later issue at <https://developer.arm.com/documentation/111167>

The product version is 1.0.

See also: [Proprietary notice](#) | [Product and document information](#) | [Useful resources](#)

Start reading

If you prefer, you can skip to [the start of the content](#).

Intended audience

This tutorial is aimed at game developers who want to apply upscaling techniques to their projects.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>.

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

Contents

1. Introduction.....	4
1.1 Before you begin.....	5
2. Setup your environment.....	6
2.1 Build the SDK.....	6
2.1.1 Integrate the SDK.....	6
2.2 Header files.....	7
2.3 Enable extensions.....	7
2.4 Padding inputs and truncating output.....	8
2.5 API interface.....	9
2.6 Reset.....	10
3. Vulkan emulation layer.....	11
4. Integration guidelines.....	13
4.1 File structure.....	13
4.2 API interface commands.....	14
4.2.1 ffxCreateContext.....	14
4.2.2 ffxDestroyContext.....	15
4.2.3 ffxConfigure.....	15
4.2.4 ffxQuery.....	16
4.2.5 ffxDispatch.....	16
4.3 Resources.....	18
Proprietary notice.....	19
Product and document information.....	21
Product status.....	21
Revision history.....	21
Conventions.....	22
Useful resources.....	24

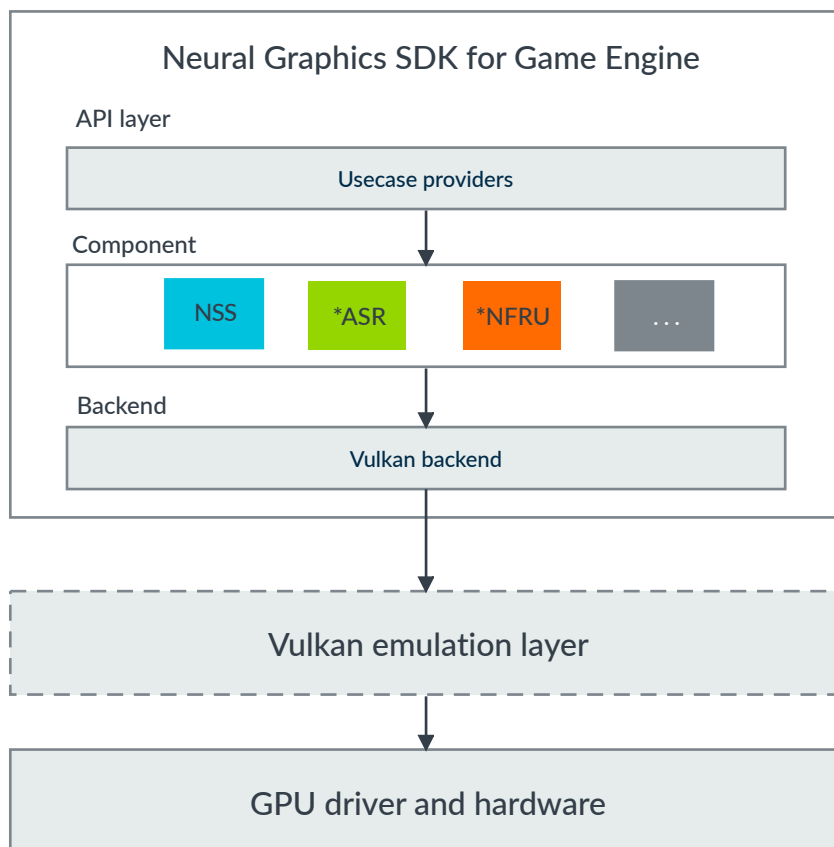
1. Introduction

The Neural Graphics SDK for Game Engines is Arm's unified graphics software development kit designed to support multiple rendering use cases across diverse engines and platforms. The Neural Graphics SDK for Game Engines is derived from AMD's FFX SDK 1.1.3.

The Neural Graphics SDK for Game Engines consolidates technologies like Neural Super Sampling (NSS), Arm Accuracy Super Resolution (ASR), and Neural Frame Rate Upsampling (NFRU) into a modular, engine-agnostic framework. Currently, the Neural Graphics SDK for Game Engines supports NSS.

NSS is Arm's next-generation neural upscaling solution, designed to deliver superior image quality and performance over traditional shader-based methods like Arm ASR. It leverages dedicated neural accelerators to enhance rendering pipelines, particularly for mobile and resource-constrained devices.

Figure 1-1: Neural Graphics SDK for Game Engines



***NFRU**

Not supported yet.

*Arm ASR

Not supported yet. But it will be a fallback for NSS because some devices cannot support ML extensions for Vulkan.

api_layer

The SDK is compliant with AMD FidelityFX API 1.1.3.

Components

Currently, NSS is supported. All components are implemented independently in this layer.

Backend

All components share one backend. The components access deeper layers using Vulkan APIs.

Vulkan emulation layer

Emulation layers for supporting the ML extensions and Vulkan headers before the real device is ready. For more information, see [Vulkan emulation layer](#).

1.1 Before you begin

This tutorial is aimed at game developers who want to apply upscaling techniques to their projects. Once you have completed the SDK integration, your project will be able to use Neural Super Sampling (NSS) for super-resolution during the post-processing stage.



The SDK is based on the Vulkan API, so we assume that readers of this document are familiar with Vulkan.

2. Setup your environment

You must have the following software versions installed:

Cmake

Minimum version 3.21, maximum version 3.31

Vulkan SDK

Recommended version 1.4.321.0

2.1 Build the SDK

This topic shows you how to build the SDK for Windows and Linux.



Note

You must only run one of the scripts depending on the operating system that you are using.

Procedure for Windows

To build the SDK in Windows, use the `build.bat` script file.

Procedure for Linux

To build the SDK in Linux, use the `build.sh` script file.

Results

If the SDK has built successfully, the libraries for SDK are generated in the `./bin` folder.

Next steps

You cannot use these libraries directly. You must integrate the SDK into a project, for example, a game engine. Some sample codes are provided in [Integration Guidelines](#).

2.1.1 Integrate the SDK

Once you have completed the SDK integration, your project will be able to use Neural Super Sampling (NSS) for super-resolution during the post-processing stage.

Procedure

1. To build from source code, copy the SDK folder into your project.
2. Add the SDK as a sub project through `CMakeLists.txt`:

```
add_subdirectory(path/to/sdk) .
```

3. If you are using prebuilt libraries, you must link them to your project.

2.2 Header files

To call the API functions, you must include header files. You must add `#define FFX_CPU` before including the header files so that the SDK can resolve some types in some common headers.

Procedure

Include the following header files:

```
#define FFX_CPU
#include <path/to/sdk/ffx-api/include/ffx_api/ffx_api.hpp>
#include <path/to/sdk/ffx-api/include/ffx_api/ffx_nss.hpp>
#include <path/to/sdk/ffx-api/include/ffx_api/ffx_api_types.h>
#include <path/to/sdk/ffx-api/include/ffx_api/vk/ffx_api_vk.hpp>
```

2.3 Enable extensions

The following topic shows you how to enable the extensions for `ARM_TENSOR` and `ARM_DATA_GRAPH` vulkan APIs.

About this task

When creating vulkan physical devices, you must enable the necessary extensions:

- `VK_ARM_tensors`
- `VK_ARM_data_graph`.

Sample code

Use the following sample code as guidance to enable your extensions:

```
VkDeviceCreateInfo ci = {};
ci.sType = VK_STRUCTURE_TYPE_DEVICE_CREATE_INFO;
... //Other create info settings

VkPhysicalDeviceTensorFeaturesARM tensorFeature = {};
{
    tensorFeature.sType = VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_TENSOR_FEATURES_ARM;

    VkPhysicalDeviceFeatures2 features2 = {};
    features2.sType = VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_FEATURES_2;
    features2.pNext = &tensorFeature;
    vkGetPhysicalDeviceFeatures2(m_physicalDevice, &features2);

    tensorFeature.pNext = const_cast<void*>(ci.pNext);
    ci.pNext = &tensorFeature;
}

VkPhysicalDeviceDataGraphFeaturesARM dataGraphFeature = {};
{
    dataGraphFeature.sType =
        VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_DATA_GRAPH_FEATURES_ARM;

    VkPhysicalDeviceFeatures2 features2 = {};
    features2.sType = VK_STRUCTURE_TYPE_PHYSICAL_DEVICE_FEATURES_2;
    features2.pNext = &dataGraphFeature;
```

```
vkGetPhysicalDeviceFeatures2(m_physicalDevice, &features2);

dataGraphFeature.pNext = const_cast<void*>(ci.pNext);
ci.pNext = &tensorFeature;
}

vkCreateDevice(m_physicalDevice, &ci, nullptr, &m_vkdevice)
```

2.4 Padding inputs and truncating output

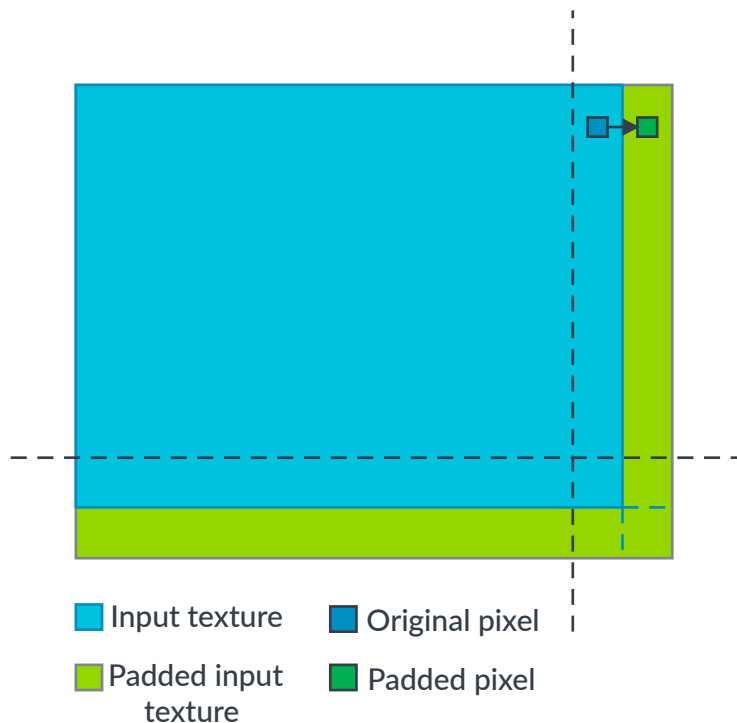
The Neural Super Sampling (NSS) model requires width and height of input textures in multiples of 8. If the input textures are not in multiples of 8, you must pad them.

Padding input

You must only pad your input if the input textures are not in multiples of 8. For example, if your input resolution is 960 * 540, you must pad it to 960 * 544.

We recommend that you use “Mirror padding” in the bottom-right corner:

Figure 2-1: Mirror padding



Sample code:

```
#version 450
```



```
layout(push_constant) uniform Constants {
    uvec2 inputSize;
    uvec2 paddedInputSize;
} g_pc;

layout(location = 0) in vec2 uv;

layout(location = 0) out vec4 outColor;
layout(location = 1) out float outDepth;
layout(location = 2) out vec2 outFlow;

layout(set = 0, binding = 0) uniform sampler2D inputColorTex;
layout(set = 0, binding = 1) uniform sampler2D inputDepthTex;
layout(set = 0, binding = 2) uniform sampler2D inputFlowTex;

void main() {
    Vec2 inputUV = (uv * Vec2(m_paddedInputSize) / Vec2(g_pc.m_inputSize));

    F32 MirrorBoundary = 1.0 - 0.5 / g_pc.m_inputSize;
    if(inputUV.x >= 1.0f)
    {
        inputUV.x = 2 * MirrorBoundary - inputUV.x;
    }
    if(inputUV.y >= 1.0f)
    {
        inputUV.y = 2 * MirrorBoundary - inputUV.y;
    }
    uv = inputUV;

    outColor = texture(inputColorTex, coord);
    outDepth = texture(inputDepthTex, coord).r;
    outFlow = motionVectorFactor * texture(inputFlowTex, coord).rg;
}
```

Truncate output

You must truncate the bottom-right corner of the output. For example, if your input dimension is 960 * 540, scale is x2, expect a 1920 * 1080 output. Then, the padded input dimension will be 960 * 544 and the padded output will be 1920 * 1088, therefore you must truncate the output to 1920 * 1080.

2.5 API interface

The SDK operates based on these APIs:

- Create context
- Destroy
- Dispatch

Create Context

1. To create context, you must prepare `ffx::CreateBackendVKDesc` and `ffx::CreateContextDescNss` first.

The `ffx::Context` data structure will then be generated.



Note

You must preserve `ffx::Context` first, before you can dispatch and destroy.

2. To create ffx context, call `ffx::CreateContext`.

For more information and sample code, see [ffxCreateContext](#).

Destroy

- When the application is terminating, or if you want to destroy the context, call `ffx::DestroyContext`.
- For more information, see [ffxDestroyContext](#).

Dispatch

- You must prepare `ffx::DispatchDescNss` for dispatch.
- To run upscale, call `ffx::Dispatch`.
- For more information and to view the sample code, see [ffxDispatch](#).

2.6 Reset

When the render resolution, or the rendered scene has changed, you must follow the steps in this section to reset the SDK's context.

Procedure

1. Destroy the old context.
2. Create a new context with a new render resolution.
3. Set `reset` in `ffx::DispatchDescNss` as true in next frame's `ffx::Dispatch`.

3. Vulkan emulation layer

After you have integrated the Neural Graphics SDK for Game Engines, you are ready to run Neural Super Sampling (NSS) in your project. If your platform does not support ML extensions for Vulkan, you must enable the Vulkan emulation layer.

Before you begin

To enable the Vulkan emulation layer, see [Vulkan emulation layer](#).

About this task

You can enable the Vulkan emulation layer in two ways:

- You can use the “Vulkan configurator” in your vulkan sdk, see [Option 1: Using the Vulkan configurator](#)
- You can add Vulkan emulation layer libs path to your system environment, see [Option 2: Using the libs path](#).

Choose one method.



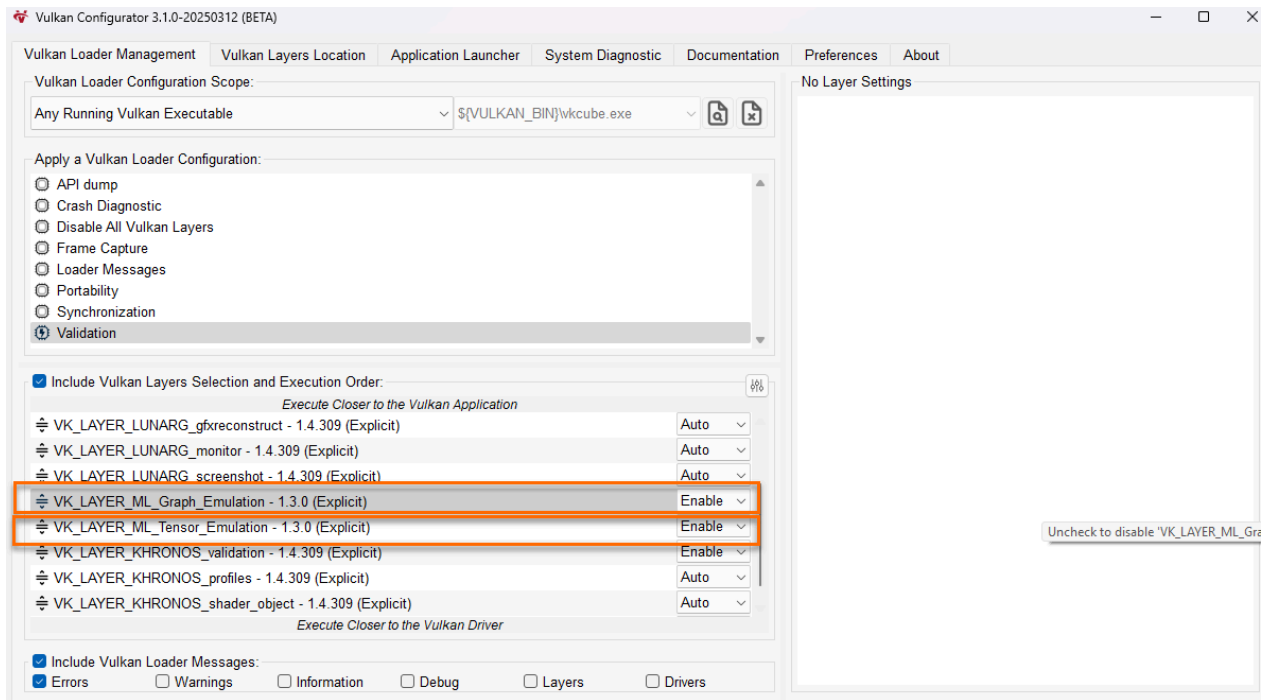
Whichever method you use, you must make sure that you enable `VK_LAYER_ML_Graph_Emulation` before you enable `VK_LAYER_ML_Tensor_Emulation`.

Option 1: Using the Vulkan configurator

This section shows you how to use the “Vulkan configurator” in your vulkan sdk to enable the Vulkan emulation layer.

1. Open the Vulkan configurator
2. Navigate to the `VK_LAYER_ML_Graph_Emulation - 1.3.0 (Explicit)` row and select **Enable** from the drop down menu.
3. Navigate to the `VK_LAYER_ML_Tensor_Emulation - 1.3.0 (Explicit)` row and select **Enable** from the drop down menu.

Figure 3-1: Vulkan emulation layer



Option 2: Using the libs path

This section shows you how to use the system environment to enable the Vulkan emulation layer.

On Windows:

```
set "VK_ADD_LAYER_PATH=path\to\VulkanML"
set "VK_INSTANCE_LAYERS=VK_LAYER_ML_Graph_Emulation;VK_LAYER_ML_Tensor_Emulation"
```

On Linux:

```
export VK_ADD_LAYER_PATH="path/to/VulkanMLLib"
export LD_LIBRARY_PATH="path/to/VulkanMLLib"
export VK_INSTANCE_LAYERS="VK_LAYER_ML_Graph_Emulation;VK_LAYER_ML_Tensor_Emulation"
```

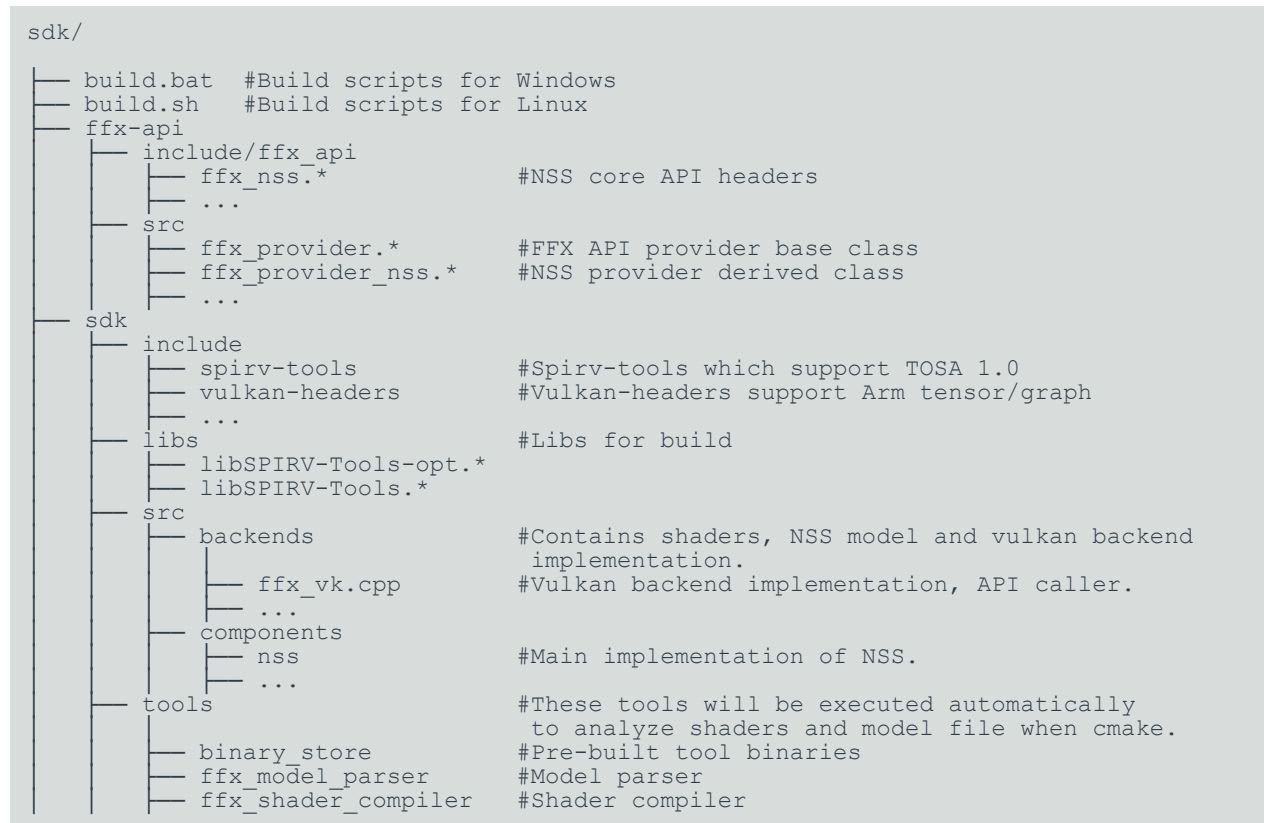
4. Integration guidelines

This section provides the best practices and rules to perform the integration. It provides more in-depth integration guidelines and technical details.

- [File structure](#)
- [API interface](#)
- [Resources](#)

4.1 File structure

The following image introduces the file structure of the Neural Graphics SDK for Game Engines. It lists main files related to Arm Neural Super Sampling (NSS).



4.2 API interface commands

The Neural Graphics SDK for Game Engines is built based on FFX SDK 1.1.3 APIs, declared in `ffx_api.h`. To learn more about the API, see the following commands in this section.

- [ffxCreateContext](#)
- [ffxDestroyContext](#)
- [ffxDispatch](#)
- [ffxQuery](#)
- [ffxConfigure](#)

4.2.1 ffxCreateContext

`ffxCreateContext` creates an FFX object context. Depending on the structures provided to this function, the context is created with the desired version and attributes.

Context generated by `ffxCreateContext` must remain live until you call `ffxDestroyContext`. If `MemCb` is null, the system allocator (malloc/free) is used instead.

Sample code:

```
ffx::CreateBackendVKDesc backendDesc{};
backendDesc.header.type = FFX_API_CREATE_CONTEXT_DESC_TYPE_BACKEND_VK;
backendDesc.vkDevice = getVulkanDevice();
backendDesc.vkPhysicalDevice = getVulkanPhysicalDevice();
backendDesc.vkInstance = getVulkanInstance();
backendDesc.vkDeviceProcAddr = vkGetDeviceProcAddr; //vulkan function pointer
backendDesc.vkGetInstanceProcAddr = vkGetInstanceProcAddr; //vulkan function pointer

ffx::CreateContextDescNss createContextNss{};
createContextNss.header.type = FFX_API_CREATE_CONTEXT_DESC_TYPE_NSS;
createContextNss.maxRenderSize = {initSettings.m_srcRes.x(),
                                   initSettings.m_srcRes.y()};
createContextNss.maxUpscaleSize = {initSettings.m_targetRes.x(),
                                    initSettings.m_targetRes.y()};
createContextNss.flags |= (FFX_API_NSS_CONTEXT_FLAG_QUANTIZED |
                           FFX_API_NSS_CONTEXT_FLAG_HIGH_DYNAMIC_RANGE |
                           FFX_API_NSS_CONTEXT_FLAG_DEPTH_INFINITE |
                           FFX_API_NSS_CONTEXT_FLAG_READ_TENSORS_AS_IMAGES |
                           FFX_API_NSS_CONTEXT_FLAG_ENABLE_DEBUG_CHECKING);
createContextNss.fpMessage = &nssMsgCallback;
ffx::ReturnCode retCode = ffx::CreateContext(m_nssContext, nullptr,
                                             createContextNss, backendDesc);
```

The following list explains the details about the `ffxApiCreateContextDesc` data structure:

header.type

Must be `FFX_API_CREATE_CONTEXT_DESC_TYPE_NSS`.

maxRenderSize/maxUpscaleSize

The SDK supports flexible upscale ratio configuration but to achieve better image quality, we recommend using an upscaling ratio below 2x. We have implemented a specialized

optimization for 2x upscaling to achieve better performance. We plan to introduce further optimizations for other ratios.

flags

These flags are defined in `enum FfxApiCreateContextNssFlags` in `sdk/ffx-api/include/ffx_api/ffx_nss.h`.

qualityMode

Select shader quality mode. This bit is currently unused. It will be required in the future when more optimizations are introduced in the shader.

fpMessage

Is a callback function, it prints a error or warning message.

The following table shows the Neural Super Sampling (NSS) context flags.

Table 4-1: Create NSS context flags

Flag Bits	Description
FFX_API_NSS_CONTEXT_FLAG_ALLOW_16BIT	Runtime allows 16 bit resources to be used.
FFX_API_NSS_CONTEXT_FLAG_DEPTH_INFINITE	Input depth buffer data provided is using an infinite far plane.
FFX_API_NSS_CONTEXT_FLAG_DEPTH_INVERTED	Input depth buffer data provided is inverted [1..0].
FFX_API_NSS_CONTEXT_FLAG_ENABLE_DEBUG_CHECKING	Runtime checks the API values and reports issues.
FFX_API_NSS_CONTEXT_FLAG_HIGH_DYNAMIC_RANGE	Input color data provided uses a high-dynamic range. Currently this flag must be set.
FFX_API_NSS_CONTEXT_FLAG_QUANTIZED	Use a quantized data graph. Resources will be quantized to 8 bits. Currently, the SDK only supports quantized data graphs, you must set this bit.
FFX_API_NSS_CONTEXT_FLAG_READ_TENSORS_AS_IMAGES	Tensor image aliasing is enabled, it loads tensors through “texture” functions.
FFX_API_NSS_CONTEXT_FLAG_RESAMPLE_BICUBIC	Sample using Bicubic filtering

4.2.2 ffxDestroyContext

Destroys an FFX object context. `memCb` must be compatible with the callbacks passed into `ffxCreateContext`.

4.2.3 ffxConfigure

`ffxConfigure` configures the provided FFX object context. If the context is null, configure operates on any global state. For Neural Super Sampling (NSS), there is no specific configuration.

4.2.4 ffxQuery

`ffxQuery` queries the provided FFX object context. Neural Super Sampling (NSS) provides these query types. The corresponding data structures for query are defined in `sdk/ffx-api/include/ffx_api/ffx_nss.h`.

Table 4-2: Query types

Query Type	Data Structure	Comments
FFX_API_QUERY_DESC_TYPE_NSS_GETJITTERPHASECOUNT	<code>ffxApiQueryDescNssGetJitterPhaseCount</code>	Gets the jitter phase count.
FFX_API_QUERY_DESC_TYPE_NSS_GETJITTEROFFSET	<code>ffxApiQueryDescNssGetJitterOffset</code>	Gets the jitter offset for the specific index.

If context is null, the query operates on any global state. For example, to query a provider ID:

```
ffx::QueryDescGetVersions versionQuery{};
versionQuery.createDescType = FFX_API_CREATE_CONTEXT_DESC_TYPE_NSS;
uint64_t versionCount = 0;
versionQuery.outputCount = &versionCount;
ffxQuery(nullptr, &versionQuery.header); //Null context pointer

std::vector<uint64_t> versionIds;
std::vector<const char*> versionNames;
versionIds.resize(versionCount);
versionNames.resize(versionCount);
versionQuery.versionIds = versionIds.data();
versionQuery.versionNames = versionNames.data();
ffxQuery(nullptr, &versionQuery.header); //Null context pointer
```

4.2.5 ffxDispatch

For `ffxDispatch` to perform the upscale work, you must call `ffxDispatch` at the end of each frame. Dispatches work on the given FFX object context defined by the dispatch descriptor. To allow upscale to work properly, you must construct the `ffxApiDispatchDescNss` data structure.

Sample code:

```
ffx::DispatchDescNss dispatchNss{};
dispatchNss.header.type = FFX_API_DISPATCH_DESC_TYPE_NSS;
dispatchNss.commandList = static_cast<CommandBufferImpl*>(cmdb).getHandle();

//Get input resources
dispatchNss.color = getColor;
dispatchNss.depth = getDepth;
dispatchNss.depthTml = getDepthTml;
dispatchNss.motionVectors = getMotion;
dispatchNss.outputTml = getOutputTml;
dispatchNss.output = getOutput;
dispatchNss.debugViews = getDebugView;

dispatchNss.jitterOffset.x = jitterOffset.x();
dispatchNss.jitterOffset.y = jitterOffset.y();
dispatchNss.renderSize.width = renderSize.x();
dispatchNss.renderSize.height = renderSize.y();
dispatchNss.upscaleSize.width = upscaleSize.x();
dispatchNss.upscaleSize.height = upscaleSize.y();
```



```
dispatchNss.cameraNear      = cameraNear;
dispatchNss.cameraFar      = cameraFar;
dispatchNss.cameraFovAngleVertical = cameraFovAngleVertical;
dispatchNss.exposure        = exposure;

//NSS expects the motion vectors in left-handed coordinates
dispatchNss.motionVectorScale.x = -1.0f * renderSize.x();
dispatchNss.motionVectorScale.y = -1.0f * renderSize.y();

dispatchNss.frameTimeDelta    = frameTimeDelta;
dispatchNss.reset             = reset;
dispatchNss.flags             = enableDebugView ?
                                FFX_API_NSS_DISPATCH_FLAG_DRAW_DEBUG_VIEW :
                                0;

ffx::ReturnCode retCode = ffx::Dispatch(m_nssContext, dispatchNss);
```

The following list explains the details about the `ffx::DispatchDescNss` data structure:

Input resources

For information on resources, see [Resources](#).

jitterOffset

Jitter offset for each frame. The jitter offsets must be in the range [-0.5, 0.5].

renderSize/upscaleSize

Same as `ffxCreatContext`.

cameraNear

The distance to the near plane of the camera.

cameraFar

The distance to the far plane of the camera.

cameraFovAngleVertical

The camera angle field of view in the vertical direction, expressed in radians.

exposure

The exposure value. The exposure value cannot be zero.

motionVectorScale

It should be set as `renderSize`. Neural Super Sampling (NSS) needs motion vectors that are recorded in the left-handed coordinate system. If the motions provided by your application are not recorded in the left-hand coordinate system, you must multiply `motionVectorScale` by -1.0.

frameTimeDelta

The time elapsed since the last frame, expressed in milliseconds.

reset

You must reset the history, due to changes in render size, camera transitions. For more information, see [Reset](#).

flags

Only contains `FFX_API_NSS_DISPATCH_FLAG_DRAW_DEBUG_VIEW`. When this flag is set, the debug view is rendered, the normal upscaled output view is not rendered.

4.3 Resources

The following table lists the details of the input and output resources of the SDK.

Table 4-3: Input and output resources

Resource	Resolution	Format	Data type	Comment
color	Render resolution	R11G11B10	Float	The render resolution color buffer for the current frame provided by the application. Must be HDR format.
depth	Render resolution	R32	Float	The render resolution depth buffer for the current frame provided by the application.
depthTm1	Render resolution	R32	Float	The last frame's depth texture. "Tm1" means "time minus 1".
motionVectors	Render resolution	R16G16	Float	The 2D motion vectors for the current frame are provided by the application. $UV(T-1) = UV(T) + \text{Motion}$. Normalized vectors, range in $[-1, 1]$.
output	Upscaled resolution	R11G11B10	Float	Upscaled output.
outputTm1	Upscaled resolution	R11G11B10	Float	The last frame's upscaled output.
debugView	Upscaled resolution	R11G11B10	Float	Render internal resources for easy debug. The debug view will split the output into 12 pieces: <ul style="list-style-type: none"> Row 1: Warped history, jittered color, feedback tensor, disocclusion mask and luma derivative. Row 2: Internal tensors, K0~K3. Row 3: Motion vector, KPN weight (calculated from internal tensors), temporal parameters, upscaledOutput.

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM 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 ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant

export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in Arm documents.

Product status

All products and services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

The information in this document is for a product under development (dev product).

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
0100-01	27 October 2025	Non-Confidential	First release for version 1.0

Change history

The Change history tables describe the technical changes between released issues of this document in reverse order. Issue numbers match the revision history in [Document release information](#) on page 21.

Table 2: Issue 0100-01

Change	Location
First limited access release for version 1.0	-

Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Terms in descriptive lists, where appropriate.
monospace	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
monospace <u>underline</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <div>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></div>
SMALL CAPITALS	Terms that have specific technical meanings as defined in the <i>Arm® Glossary</i> . For example, IMPLEMENTATION DEFINED , IMPLEMENTATION SPECIFIC , UNKNOWN , and UNPREDICTABLE .



We recommend the following. If you do not follow these recommendations your system might not work.



Your system requires the following. If you do not follow these requirements your system will not work.



You are at risk of causing permanent damage to your system or your equipment, or harming yourself.



This information is important and needs your attention.



A useful tip that might make it easier, better or faster to perform a task.



A reminder of something important that relates to the information you are reading.

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Arm documents are available on developer.arm.com/documentation.

Confidential documents are only available to licensees, when logged in. Each document link in the tables below provides direct access to the online version of the document.

Arm product resources	Document ID	Confidentiality
Arm® Moving Mobile Graphics	Moving Mobile Graphics	Non-Confidential
Arm® Neural Graphics Developer Tools Quick Start Guide	110291	Confidential