

[Accelerate](#) / [BNNSGraph](#) / BNNSGraph.Context

Class

BNNSGraph.Context

A wrapper around a compiled graph object that adds a required modifiable context to support dynamically sized models and set execute-time options.

iOS 18.0+ | iPadOS 18.0+ | Mac Catalyst | macOS 15.0+ | tvOS 18.0+ | visionOS | watchOS 11.0+

```
class Context
```

Overview

A `BNNSGraph.Context` instance provides a wrapper around the C API `bnns_graph_t` and `bnns_graph_context_t` types. Because this class manages its own memory, you don't need to call `BNNSGraphContextDestroy(_:_)` to deallocate its resources.

Topics

Creating a graph context

```
struct CompileOptions
```

The compilation options that BNNS uses when compiling a source `mlmodelc` file to a graph object.

Specifying and querying a graph context's properties

```
func setDynamicShapes([BNNSGraph.Shape], forFunction: String?) async  
throws -> [BNNSGraph.Shape]
```

Specifies the dynamic shapes for a graph and, if possible, infers the output shapes.

```
struct Shape
```

The specification of the shape of an argument.

```
func argumentCount(forFunction: String?) -> Int
```

Returns the number of arguments for the given function argument.

```
func argumentNames(forFunction: String?) -> [String]
```

Returns the names of arguments for the given function argument.

```
func argumentPosition(forFunction: String?, argument: String) -> Int
```

Returns the index into the arguments array for the given function argument.

```
var functionCount: Int
```

The number of input arguments for the given function argument.

```
var functionNames: [String]
```

Returns the names of callable functions in the graph.

```
var checkForNaNsAndInfinities: Bool
```

A Boolean value that specifies that the context checks intermediate tensors for NaNs and infinities.

Specifying a tensor's properties

```
func tensor(forFunction: String?, argument: String, fillKnownDynamicShapes: Bool) -> BNNTensor?
```

Returns an unallocated tensor for a given function argument.

Executing a graph

```
func executeFunction(String?, arguments: inout [BNNTensor]) async throws
```

Executes the specified function using an array of input and output tensors.

```
func executeFunction<T>(String?, arguments: [T]) throws
```

Executes the specified function using an array of input and output pointers.

```
protocol PointerArgument
```

A type that BNNS Graph accepts as an input-output argument.

Handling errors

```
enum Error
```

Error codes that a graph context throws.

Initializers

```
init(compileFromPath: String, functionName: String?, options: BNNSGraph.CompileOptions) async throws
```

Returns a new context that wraps a graph object which represents the compiled .mlmodelc file.

```
init(compileFromPath: String, functionName: String?, options: BNNSGraph.CompileOptions) throws
```

Synchronously returns a new context that wraps a graph object which represents the compiled .mlmodelc file.

Instance Properties

```
var streamingAdvanceCount: Int
```

Sets streaming advancement amount for cases with dynamically shaped inputs.

Instance Methods

```
func allocateTensor(forFunction: String?, argument: String, fillKnownDynamicShapes: Bool) -> BNNSTensor?
```

Returns an allocated tensor for a given function argument.

```
func executeFunction(String?, arguments: inout [BNNSTensor]) throws
```

Synchronously executes the specified function with the provided context.

```
func setBatchSize(Int, forFunction: String?) async
```

Sets the batch size for a graph.

```
func setBatchSize(Int, forFunction: String?)
```

Synchronously sets the batch size for a graph.