# callAppShutdownHook

August 17, 2026

## callAppShutdownHook

**Kind:** Function

**Source:** `packages/core/hooks/on-app-shutdown.hook.ts`

**Part of:** Core

Calls the `onApplicationShutdown` function on the module and its children (providers / controllers).

`callAppShutdownHook` orchestrates application shutdown lifecycle handling for a module. It invokes `onApplicationShutdown` on the module itself and on eligible providers, controllers, injectables, and middleware instances, forwarding an optional shutdown signal such as `SIGTERM`.

### Signature

```
async function callAppShutdownHook(module: Module, signal: string): Promise<any>
```

### Parameters

| Name | Type |
| --- | --- |
| `module` | `Module` |
| `signal` | `string` |

**Returns:** `Promise<any>`

### Diagram

```
graph LR
  A[Application shutdown signal] --> B[callAppShutdownHook]
  B --> C[Resolve module providers and children]
```

```
  C --> D[Invoke non-transient instances]
  D --> E[Invoke transient instances]
  E --> F[onApplicationShutdown(signal)]
```

## Usage

```
import { callAppShutdownHook } from '@nestjs/core/hooks/on-app-shutdown.hook';

// Typically called by the Nest application shutdown lifecycle internally.
async function shutdownModule(moduleRef: any) {
  await callAppShutdownHook(moduleRef, 'SIGTERM');

  console.log('Module shutdown hooks completed.');
}
```

## AI Coding Instructions

- Treat this as framework lifecycle infrastructure; application code should normally implement `onApplicationShutdown` rather than call this function directly.
- Pass the operating-system shutdown signal when available so lifecycle hooks can distinguish graceful termination causes.
- Preserve the shutdown ordering between non-transient and transient instances when modifying hook execution behavior.
- Ensure providers, controllers, middleware, and module instances with `onApplicationShutdown` remain discoverable through the module container.

## Relationships

- IMPORTS → `OnApplicationShutdown`
- IMPORTS → `isFunction`
- IMPORTS → `isNil`