SerializedGraphMetadata

August 18, 2026

SerializedGraphMetadata

Kind: Interface

Source: `packages/core/inspector/interfaces/serialized-graph-metadata.interface.ts`

Part of: Core

`SerializedGraphMetadata` describes diagnostic metadata attached to a serialized graph when graph construction or dependency resolution encounters a problem. Its `cause` field identifies the failure category and can include dependency context, module/node identifiers, and the original error for inspection tooling.

Properties

Property	Type
`cause`	`{ type: 'unknown-dependencies'

Diagram

```
graph LR
  Graph[Serialized Graph] --> Metadata[SerializedGraphMetadata]
  Metadata --> Cause[cause]
  Cause --> Type["type: unknown-dependencies | unknown"]
  Cause --> Context["context?: InjectorDependencyContext"]
  Cause --> Module["moduleId?: string"]
  Cause --> Node["nodeId?: string"]
  Cause --> Error["error?: any"]
```

Usage

```
import type { SerializedGraphMetadata } from '@nestjs/core/inspector/interfaces/serialized-grap

const metadata: SerializedGraphMetadata = {
```

```
  cause: {
    type: 'unknown-dependencies',
    moduleId: 'AppModule',
    nodeId: 'UsersService',
    error: new Error('Unable to resolve dependency DatabaseService'),
  },
};

// Attach metadata to serialized graph output for inspector/debugging consumers.
console.error(metadata.cause.type, metadata.cause.error);
```

AI Coding Instructions

- Set `cause.type` to `'unknown-dependencies'` when dependency injection resolution fails; use `'unknown'` for uncategorized graph failures.
- Include `moduleId` and `nodeId` whenever they are available to make inspector output traceable.
- Preserve the original thrown value in `error` rather than replacing it with a string.
- Provide `context` only when dependency resolution context is available from the injector.
- Treat all fields other than `cause.type` as optional when consuming serialized graph metadata.

How it works

`SerializedGraphMetadata` is a TypeScript interface for the optional `metadata` field of a serialized graph JSON object. That graph field is optional, while `cause` is required whenever metadata is present. serialized-graph-json.interface.ts:8-14 serialized-graph-metadata.interface.ts:3-10

Its required `cause.type` is restricted to:

- `'unknown-dependencies'`
- `'unknown'`
  serialized-graph-metadata.interface.ts:4-5

The `cause` object may also contain:

- `context` : an `InjectorDependencyContext` , whose optional fields identify a property key, injection name/token, dependency index, or dependency array. serialized-graph-metadata.interface.ts:6 injector.ts:67-85
- `moduleId` : a string. serialized-graph-metadata.interface.ts:7

- `nodeId`: a string. serialized-graph-metadata.interface.ts:8
- `error`: an `any`-typed value. serialized-graph-metadata.interface.ts:9

During dependency-instance creation, if `createInstances` throws, `InstanceLoader` first inspects the modules, registers a partial graph with the caught error, then rethrows that error. instance-loader.ts:25-38 `GraphInspector.registerPartial` sets the graph status to `'partial'`. graph-inspector.ts:38-40

For an `UnknownDependenciesException`, it writes metadata with `type: 'unknown-dependencies'`, copies the exception's dependency context, and conditionally records the module and node IDs. graph-inspector.ts:41-49 The exception exposes the context as required constructor input, while its module reference and node metadata are optional. unknown-dependencies.exception.ts:6-17

For any other caught value, it writes `type: 'unknown'` and stores that value in `cause.error`; it does not populate `context`, `moduleId`, or `nodeId` in this branch. graph-inspector.ts:50-57

`SerializedGraph` stores this interface value through its `metadata` setter and includes it as `metadata` in `toJSON()` only when metadata has been assigned. serialized-graph.ts:35 serialized-graph.ts:56-58 serialized-graph.ts:125-139 The partial graph is then registered in `PartialGraphHost`, whose `toJSON()` delegates to the stored graph's `toJSON()`. graph-inspector.ts:58 partial-graph.host.ts:3-16