PartialGraphHost

August 17, 2026

PartialGraphHost

Kind: Class

Source: `packages/core/inspector/partial-graph.host.ts`

Part of: Core

`PartialGraphHost` is a static host for storing and exposing the most recently generated partial dependency graph. It delegates serialization to a `SerializedGraph`, allowing inspector and error-reporting flows to retrieve the graph as JSON or a formatted string after it has been registered.

Methods

Method	Signature	Returns
`toJSON`	`toJSON()`	`void`
`toString`	`toString()`	`void`
`register`	`register(partialGraph: SerializedGraph)`	`void`

Diagram

```
graph LR
  A[Graph inspection or bootstrap failure] --> B[SerializedGraph]
  B -->|register(graph)| C[PartialGraphHost]
  C -->|toJSON()| D[Structured graph data]
  C -->|toString()| E[Human-readable graph output]
```

Usage

```
import { PartialGraphHost } from '@nestjs/core/inspector/partial-graph.host';
import { SerializedGraph } from '@nestjs/core/inspector/serialized-graph';
```

```
// Typically produced by the inspector while scanning application modules.
const partialGraph = new SerializedGraph();

// Store the graph for later diagnostics or reporting.
PartialGraphHost.register(partialGraph);

// Retrieve serialized output.
const graphJson = PartialGraphHost.toJSON();
const graphText = PartialGraphHost.toString();

console.log(graphJson);
console.log(graphText);
```

AI Coding Instructions

- Treat `PartialGraphHost` as a static global holder; do not instantiate it or rely on per-instance state.
- Register a complete `SerializedGraph` before calling `toJSON()` or `toString()` to avoid reporting stale graph data.
- Keep serialization logic in `SerializedGraph`; `PartialGraphHost` should only store and delegate to the registered graph.
- Use this host for inspector, diagnostics, and bootstrap-error integration points where a partial dependency graph must remain accessible.
- Be aware that calling `register()` replaces the previously stored graph for the entire process.

How it works

- `PartialGraphHost` is an exported class with one private, class-level `SerializedGraph` reference named `partialGraph`. [packages/core/inspector/partial-graph.host.ts:3-4]
- `register(partialGraph)` assigns its `SerializedGraph` argument to that shared reference. A later call replaces the previously stored graph; the method contains no visible validation or error handling. [packages/core/inspector/partial-graph.host.ts:14-16]
- `toJSON()` optionally calls `toJSON()` on the stored graph. If no graph has been registered, optional chaining causes it to return `undefined`; it does not throw from this method. [packages/core/inspector/partial-graph.host.ts:6-8] When a graph exists, the delegated serialization contains object forms of its nodes, edges, and entrypoints, plus extras and, when set, status and metadata. [packages/core/inspector/serialized-graph.ts:125-140]

- `toString()` likewise optionally delegates to the stored graph. A registered `SerializedGraph` formats its JSON with two-space indentation, converts symbols to strings, and serializes functions as their name or `"Function"`. [packages/core/inspector/partial-graph.host.ts:10-12] [packages/core/inspector/serialized-graph.ts:142-150]
- The in-repository registration path is `GraphInspector.registerPartial(error)`: it marks its graph's status as `'partial'`, records metadata for either an `UnknownDependenciesException` or an unknown error, then registers that graph with this host. [packages/core/inspector/graph-inspector.ts:38-59]