



CORE

DeterministicUuidRegistry

Kind: Class

VERSION
47

UPDATED
7 August 2026

DOCUMENT
reference-subsystem-packages-core/class-deterministicuuidregistry

CONTENTS

1	Methods	2	5	AI Coding Instructions	3
2	Where it refuses work	2	6	How it works	3
3	Diagram	2	7	DeterministicUuidRegistry	3
4	Usage	3			

Kind: Class

Source: packages/core/inspector/deterministic-uuid-registry.ts

Part of: Core

`DeterministicUuidRegistry` maintains stable UUID values for identifiers encountered during inspection. It returns the same UUID for a given registry key throughout a registry lifecycle and can reset its stored mappings with `clear()`.

1 Methods

Method	Signature	Returns
<code>get</code>	<code>get(str: string, inc: undefined)</code>	<code>void</code>
<code>clear</code>	<code>clear()</code>	<code>void</code>

2 Where it refuses work

- `DeterministicUuidRegistry` stops the work with an early return when `this.registry.has(id)`.

3 Diagram

```
graph LR
    A[Inspector / Caller] -->|get(stable key)| B[DeterministicUuidRegistry]
    B -->|existing mapping| C[Previously assigned UUID]
    B -->|new mapping| D[Deterministically generated UUID]
    D --> E[Registry storage]
    E --> C
    A -->|clear()| B
    B -->|remove mappings| E
```

4 Usage

```
import { DeterministicUuidRegistry } from "../deterministic-uuid-registry";

const uuidRegistry = new DeterministicUuidRegistry();

// Repeated lookups for the same stable key return the same UUID.
const firstId = uuidRegistry.get("workflow-step:validate-input");
const secondId = uuidRegistry.get("workflow-step:validate-input");

console.log(firstId === secondId); // true

// Reset mappings when starting a new inspection lifecycle.
uuidRegistry.clear();
```

5 AI Coding Instructions

- Use stable, meaningful keys when calling `get()` so generated UUIDs remain consistent across related inspection operations.
- Reuse a single registry instance for the full inspection or serialization lifecycle that requires stable identifiers.
- Call `clear()` only when beginning a new independent lifecycle; clearing during processing invalidates existing mappings.
- Do not treat returned UUIDs as permanent persisted IDs unless the registry lifecycle and input keys are also persisted.

6 How it works

7 `DeterministicUuidRegistry`

`DeterministicUuidRegistry` is a static registry that generates string IDs from an input string and tracks IDs already returned during the registry's current lifetime. Its state is a private static `Map<string, boolean>`, shared by every call to the class. [packages/core/inspector/deterministic-uuid-registry.ts:1-2]