SettlementSignal

August 17, 2026

SettlementSignal

Kind: Class

Source: `packages/core/injector/settlement-signal.ts`

Part of: Core

SettlementSignal is used to signal the resolution of a provider/instance.
Calling `complete` or `error` will resolve the promise returned by `asPromise`.
Can be used to detect circular dependencies.

`SettlementSignal` coordinates the lifecycle of a provider or instance while it is being resolved by the injector. Calling `complete()` or `error()` settles the promise returned by `asPromise()`, while reference tracking through `insertRef()` and `isCycle()` helps identify circular dependencies.

Methods

Method	Signature	Returns
`complete`	`complete()`	`void`
`error`	`error(err: unknown)`	`void`
`asPromise`	`asPromise()`	`void`
`insertRef`	`insertRef(wrapperId: string)`	`void`
`isCycle`	`isCycle(wrapperId: string)`	`void`

Diagram

```
graph LR
  Injector[Injector resolving provider] --> Signal[SettlementSignal]
  Signal --> Promise[asPromise()]
  Signal -->|complete()| Resolved[Provider resolved]
  Signal -->|error()| Failed[Provider resolution failed]
```

```
  Signal -->|insertRef()| References[Resolution references]
  References -->|isCycle()| Cycle[Circular dependency detected]
```

Usage

```
import { SettlementSignal } from "./settlement-signal";

async function resolveProvider() {
  const signal = new SettlementSignal();

  const resolution = signal.asPromise();

  try {
    // Create and initialize the provider instance.
    const instance = { connected: true };

    signal.complete();

    await resolution;
    return instance;
  } catch (error) {
    signal.error();
    throw error;
  }
}
```

AI Coding Instructions

- Create a `SettlementSignal` for each in-progress provider or instance resolution that must be observed asynchronously.
- Call `complete()` only after the provider has been fully constructed and initialized.
- Call `error()` when resolution fails so consumers awaiting `asPromise()` are unblocked.
- Use `insertRef()` while traversing dependency references, then check `isCycle()` before continuing resolution.
- Avoid awaiting `asPromise()` from a dependency path that has already been identified as circular.