

Username: Pralay Patoria **Book:** Under the Hood of .NET Memory Management. No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

Runtime GC Latency Control

.NET Framework 3.5.1 allows the latency of the GC to be controlled programmatically, which ultimately overrides the `gcConcurrent` setting within your `config` file.

To achieve this, the `System.Runtime.GCSettings.LatencyMode` property can be set to one of three modes.

- **`GCConcurrencyMode.Batch`** – designed to give maximum throughput and performance for sections of an app where UI responsiveness isn't important.
- **`GCConcurrencyMode.LowLatency`** – this mode reduces the impact of GC to a minimum, which is ideal for times when things like UI responsiveness are critical, e.g. animation.
- **`GCConcurrencyMode.Interactive`** – Workstation GC with Concurrency switched on, giving a balance between GC efficiency and app responsiveness.

An obvious use of `LatencyMode` is to change it for a short period during execution of critical code that needs maximum UI or batch processing performance, and then change it back on completion.

```
using System.Runtime;
...
// Store current latency mode
GCConcurrencyMode mode = GCSettings.LatencyMode;
// Set low latency mode
GCSettings.LatencyMode = GCConcurrencyMode.LowLatency;
try
{
    // Do some critical animation work
}
finally
{
    // Restore latency mode
    GCSettings.LatencyMode = mode;
}
```

Listing 3.4: Using GC `LatencyMode`.