Asynchronous Exit Points for .NET

Asynchronous programming patterns can enhance the scalability and performance of applications. Microsoft .NET lets you designate methods as asynchronous tasks.

The .NET runtime releases resources for asynchronous methods while tasks complete. When task processing finishes, the runtime calls back to the originating asynchronous method so the method may continue processing.

The .NET Agent detects certain asynchronous programming patterns as exit points. Because tasks may execute in parallel, AppDynamics sometimes represents asynchronous activity differently from synchronous activity in the Controller.

AppDynamics differentiates .NET async backend tracking from thread correlation. To configure thread correlation, see Thread Correlation for .NET.

Supported Asynchronous Exit Point Patterns

The agent discovers exit points for Microsoft .NET 4.5 `async` and `await` keywords. See Asynchronous Programming with Async and Await.

The agent detects asynchronous exit points for the following backend types:

- ADO.NET
- Azure Blob Storage, Azure File Storage, and Azure Table Storage
- HTTP
- MongoDB
- WCF
Flow maps in the Controller UI represent asynchronous exit points as dotted lines labeled "async". Because calls in an asynchronous transaction may execute simultaneously, the Controller doesn't display end-to-end percentage times for individual asynchronous calls.

Troubleshoot Asynchronous Calls in Transaction Snapshots

Transaction snapshots include several features to help you discover problem areas in business transactions that use asynchronous methods. For an overview of transaction snapshots, see Transaction Snapshots.

Transaction Snapshot Flow Map

The Transaction Snapshot Flow Map graphically represents the business transaction. It displays the user experience, execution time, and timestamp of the transaction. The flow map also provides details of the overall time that is spent in a particular tier and in database and remote service calls. The `async` label indicates asynchronous calls.

Snapshot Execution Waterfall View

The transaction Snapshot Execution Waterfall View shows a timeline representation of the end-to-end execution of the business transaction. Synchronous and asynchronous processes appear on a bar diagram illustrating their relative execution time arranged in chronological order. Synchronous calls are represented by solid line arrows, and asynchronous calls are represented by dotted line arrows.

The waterfall view enables you to visually identify which processes are running the longest. Double-click a bar to drill down to a call graph and investigate problematic code.

Analyze Asynchronous Activity in the Metric Browser

The Metric Browser displays asynchronous activity in the following places:

- Business Transaction Performance > Business Transactions > tier > business transaction > Thread Tasks > Asynchronous Operation > External Calls
- Overall Application Performance > tier > Thread Tasks > Asynchronous Operation > External Calls
- Overall Application Performance > tier > Individual Nodes > node name > Thread Tasks > Asynchronous Operation > External Calls

Backend types are different for the .NET Core microservices agent. See .NET Core Microservices Agent Support.