

# Blocking Session Wait Time (Seconds) alert

The Blocking Session Wait Time (Seconds) alert provides the time in seconds that a SQL Server session blocks other sessions.

By default, locked sessions waiting for a lock wait forever, which is not optimum behavior from an end-user point of view. You can change this default behavior by adding `SET LOCKTIMEOUT 60000` to the top of the first T-SQL batch after making a connection.

The [Active Alerts view](#) provides additional information on the Blocking Session Wait Time (Seconds) alert. To access this information, refer to the right-click context menu after having organized your alert view by metric. This menu grants access to real-time alert information, historical view, block details, and online help specific to the alert.



To access similar information on the [Blocking view](#), select a specific SQL Server instance, point to Sessions, and then select the Blocking ribbon.

## Blocking Sessions Report

The Blocking Sessions Report displays blocking and blocked sessions' information. To access the Blocking Sessions Report in the Alert view, select the **Show Block Details** option in the right-click context menu.

Key information provided for blocking and blocked sessions includes:

Blocking Process Details:

- Session ID
- Hostname
- User Name
- Application
- Database
- Last Batch Started
- Transaction ID
- Open Transactions

Blocked Process Details:

- Session ID
- Hostname
- User Name
- Application
- Database
- Last Batch Started
- Transaction ID
- Open Transactions
- Wait Time (ms)
- Wait Resource



To retrieve the blocking sessions report for SQL Diagnostic Manager, select the **"Non-Query activities>Capture Blocking (SQL 2005+)"** check box in the Activity Monitor Tab.

## Reduce blocked sessions wait time

If you experience more than one blocked session on an average day, your site may experience one of the following issues:

- T-SQL batches submitted containing a `BEGIN TRANSACTION` statement with no corresponding `COMMIT TRANSACTION` statement. You must correct the T-SQL.
- T-SQL batches submitted containing a `BEGIN TRANSACTION` statement but where the `COMMIT TRANSACTION` statement is in a following T-SQL batch that is only executed once the end-user confirms the transaction. You must correct the T-SQL.
- The site includes some long-running transactions processing at peak times.

Where your transactions run longer than they should and clash, consider drastically reducing the time that a lock is held by:

- Performing as much work as possible before the transaction performs its first update, delete, or insert. For example, add any necessary `SELECT` statements.
- Grouping all `UPDATES`, `DELETES`, and `INSERTS` as closely as possible within a transaction with as few `SELECTS` as possible separating them.
- Committing the transaction as soon as possible after the final DML statement.

- Avoiding any stops for user input once the transaction begins. Be sure to gather all user inputs before the transaction starts.
- Avoiding the use of server-side cursors during a transaction as they slow execution considerably.
- Minimizing or eliminating the number of SQL re-compilations made to the object if a stored procedure and/or trigger is invoked inside a transaction. See the SQL RE-compilations counter for steps to dramatically reduce recompiles.
- Increasing the speed of transaction throughput such that it becomes less likely that one transaction waits for the preceding transaction. You can improve transaction throughput speed by:
  - Adding more disks to your RAID solution.
  - Replacing your disks with faster disks.
  - Switching your RAID array from a RAID 5 to a RAID 10 solution. Note that each write IO results in two writes using RAID 10 vs. four for RAID 5 (100% more efficient with writes).
  - Switching the RAID controller cache mode from Write-through to Write-back so long as the RAID controller has some form of battery backup. This change increases the system's ability to handle write IOs by an order of magnitude.
  - Adding more cache memory to the RAID controller.
  - Adding more RAM to the server.
  - Adding another CPU to the SMP computer.
  - Upgrading the CPU, memory, and motherboard with faster models.
  - Minimizing the number of Context Switches by turning on Use NT Fibers in SQL Server.
  - Switching on the Boost SQL Server Priority.



## Create an alert response bundle

Create an alert response bundle with the Blocking Session Wait Time (Seconds) alert and related alerts. For additional information, see [Create alert response bundles](#).