


# sp\_server\_diagnostics (Transact-SQL)

14.11.2017 • 4 Minuten Lesedauer • Beitragende  

## In diesem Artikel

[Syntax](#)

[Argumente](#)

[Rückgabecodewerte](#)

[Resultsets](#)

[Hinweise](#)

[Berechtigungen](#)

[Beispiele](#)

[Siehe auch](#)


**GILT FÜR:**  SQL Server (ab 2012)  Azure SQL-Datenbank  Azure SQL Data Warehouse  Parallel Data Warehouse

Erfasst Diagnosedaten und Zustandsinformationen zu SQL Server, um potenzielle Fehler zu erkennen. Die Prozedur im Wiederholungsmodus ausgeführt und sendet regelmäßig Ergebnisse. Sie kann über eine reguläre oder eine DAC-Verbindung aufgerufen werden.

**Gilt für:** SQL Server (SQL Server 2012 (11.x) bis SQL Server 2017).

 [Transact-SQL Syntax Conventions \(Transact-SQL-Syntaxkonventionen\)](#).

## Syntax

	 Kopieren
<pre>sp_server_diagnostics [@repeat_interval =] 'repeat_interval_in_seconds'</pre>	

## Argumente

[ @repeat\_interval = ] 'repeat\_interval\_in\_seconds' Gibt das Zeitintervall, in dem die gespeicherte Prozedur wiederholt ausgeführt wird, um Zustandsinformationen zu senden.

*Repeat\_interval\_in\_seconds* ist **Int** hat den Standardwert 0. Die gültigen Parameterwerte sind 0 sowie alle Werte größer oder gleich 5. Die gespeicherte Prozedur muss mindestens 5 Sekunden lang ausgeführt werden, um vollständige Daten zurückzugeben. Der minimale Wert für die Ausführung der gespeicherten Prozedur im Wiederholungsmodus beträgt 5 Sekunden.

Wenn dieser Parameter nicht angegeben ist oder der angegebene Wert 0 beträgt, gibt die gespeicherte Prozedur einmal Daten zurück und wird dann beendet.

Wenn der angegebene Wert kleiner als der minimale Wert ist, wird ein Fehler ausgelöst und kein Wert zurückgegeben.

Wenn der angegebene Wert größer oder gleich 5 ist, wird die gespeicherte Prozedur wiederholt ausgeführt, um den Zustand zurückzugeben, bis sie manuell abgebrochen wird.

## Rückgabecodewerte

0 (Erfolg) oder 1 (Fehler)

## Resultsets

**Sp\_server\_diagnostics** die folgenden Informationen zurückgegeben

Spalte	Datentyp	Description
creation_time	datetime	Gibt den Zeitstempel der Zeilenerstellung an. Jede Zeile in einem einzelnen Rowset weist denselben Zeitstempel auf.
component_type	sysname	Gibt an, ob die Zeile Informationen für enthält die SQL Server -Instanzebene Komponente, oder für eine Always On-verfügbarkeitsgruppe:  Instanz  AlwaysOn: AvailabilityGroup
component_name	sysname	Gibt den Namen der Komponente oder den Namen der Verfügbarkeitsgruppe an:  System  resource  query_processing  io_subsystem  -Ereignisse  <Name der verfügbarkeitsgruppe >
state	int	Gibt den Integritätsstatus der Komponente an:  0  1  2  3
state_desc	sysname	Beschreibt die Zustandsspalte. Folgende Beschreibungen entsprechen den Werten in der Statusspalte:  0: Unknown  1: clean  2: Warnung  3: Fehler
data	varchar (max)	Gibt Daten an, die für die Komponente spezifisch sind.

Im Folgenden finden Sie die Beschreibungen der fünf Komponenten:

- **system:** Erfasst Daten von einer Systemperspektive Spinlocks, verarbeitungsbedingungen, offenbar keine Tasks, Seitenfehlern und CPU-Auslastung. Diese Informationen ergeben eine allgemeine Empfehlung zum Integritätsstatus.
- **resource:** Sammelt Daten aus ressourcenperspektive auf physischen und virtuellen Arbeitsspeichers, Pufferpools, Seiten, Cache und anderen Arbeitsspeicherobjekten. Diese Informationen erzeugt eine allgemeine Empfehlung zum Integritätsstatus.
- **Query\_processing:** Sammelt Daten hinsichtlich der Verarbeitung von Abfragen auf die Anzahl der Arbeitsthreads, Tasks, Wartetypen, CPU-intensiven Sitzungen und blockierenden Tasks an. Diese Informationen erzeugt eine allgemeine Empfehlung zum Integritätsstatus.
- **io\_subsystem:** Erfasst Daten zu EA. Zusätzlich zu den Diagnosedaten erzeugt diese Komponente nur für ein EA-Subsystem einen komplett fehlerfreien oder einen Warnzustand

Das System einen komplett korrekten oder einen Warnzustand.

- **Ereignisse:** Sammelt Daten und Oberflächen, die über die gespeicherte Prozedur, auf die Fehler und Ereignisse von Interesse, die von dem Server, einschließlich Details zu ringpufferausnahmen, speicherbroker, ungenügender Arbeitsspeicher, Zeitplanungsmodul-Überwachung, Pufferpool, Spinlocks Ring Buffer Ereignisse aufgezeichnet wurden, Sicherheit und Konnektivität. Ereignisse zeigen als Status immer 0 an.
- **<Name der Verfügbarkeitsgruppe >:** Sammelt Daten für die angegebene Verfügbarkeitsgruppe (wenn Component\_type = "immer auf: AvailabilityGroup").

## Hinweise

Die Komponenten system, resource und query\_processing werden zur Fehlererkennung aus Fehlerperspektive genutzt, während die Komponenten io\_subsystem und events nur zu Diagnosezwecken genutzt werden.

In der folgenden Tabelle sind die Komponenten den jeweils zugeordneten Integritätszuständen zugeordnet.

Komponenten	Clean (1)	Warning (2)	Error (3)	Unknowns (0)
System	x	x	x	
resource	x	x	x	
query_processing	x	x	x	
io_subsystem	x	x		
-Ereignisse				x

Das (x) in jeder Zeile steht für gültige Zustände für die Komponente. Im Beispiel wird io\_subsystem als fehlerfrei oder Warnung angezeigt. Der Fehlerstatus wird nicht angezeigt.

### ⓘ Hinweis


Ausführung der internen Sp\_server\_diagnostics-Prozedur ist für einen präemptiven Thread mit hoher Priorität implementiert.

## Berechtigungen

Erfordert die VIEW SERVER STATE-Berechtigung auf dem Server.

## Beispiele

Es ist empfehlenswert, die Zustandsinformationen in erweiterten Sitzungen aufzuzeichnen und in einer Datei zu speichern, die sich außerhalb von SQL Server befindet. In diesem Fall können Sie auch bei einem Fehler auf diese zugreifen. Im folgenden Beispiel wird die Ausgabe einer Ereignissitzung in einer Datei gespeichert:

SQL	 Kopieren
<pre>CREATE EVENT SESSION [diag] ON SERVER     ADD EVENT [sp_server_diagnostics_component_result] (set collect_data=1)     ADD TARGET [asynchronous_file_target] (set filename='c:\temp\diag.xel'); GO ALTER EVENT SESSION [diag] ON SERVER STATE = start; GO</pre>	

In der unten angegebenen Beispielabfrage wird die Protokolldatei der erweiterten Sitzung gelesen:

SQL	 Kopieren
-----	--

```

SELECT
    xml_data.value('/event/@name')[1], 'varchar(max)') AS Name
, xml_data.value('/event/@package')[1], 'varchar(max)') AS Package
, xml_data.value('/event/@timestamp')[1], 'datetime') AS 'Time'
, xml_data.value('/event/data[@name='component_type']/value)[1], 'sysname') AS Sysname
, xml_data.value('/event/data[@name='component_name']/value)[1], 'sysname') AS Component
, xml_data.value('/event/data[@name='state']/value)[1], 'int') AS State
, xml_data.value('/event/data[@name='state_desc']/value)[1], 'sysname') AS State_desc
, xml_data.query('/event/data[@name="data"]/value/*') AS Data
FROM
(
    SELECT
        object_name as event
        , CONVERT(xml, event_data) as xml_data
    FROM
        sys.fn_xe_file_target_read_file('C:\Program Files\Microsoft SQL
Server\MSSQL13.MSSQLSERVER\MSSQL\Log\*.xel', NULL, NULL, NULL)
)
AS XEventData
ORDER BY time;

```

Im folgenden Beispiel wird die Ausgabe von sp\_server\_diagnostics an eine Tabelle in einem anderen als dem Wiederholungsmodus aufgezeichnet:

SQL	Kopieren
<pre> CREATE TABLE SpServerDiagnosticsResult (     create_time DateTime,     component_type sysname,     component_name sysname,     state int,     state_desc sysname,     data xml ); INSERT INTO SpServerDiagnosticsResult EXEC sp_server_diagnostics; </pre>	

Die nachstehende Beispielabfrage liest die Zusammenfassungsausgabe aus der Tabelle:

SQL	Kopieren
<pre> SELECT create_time,        component_name,        state_desc FROM SpServerDiagnosticsResult; </pre>	

Die nachstehende Beispielabfrage liest einige Bestandteile der ausführlichen Ausgabe aus jeder Komponente in der Tabelle:

SQL	Kopieren
<pre> -- system select data.value('/system/@systemCpuUtilization')[1], 'bigint') as 'System_CPU',        data.value('/system/@sqlCpuUtilization')[1], 'bigint') as 'SQL_CPU',        data.value('/system/@nonYieldingTasksReported')[1], 'bigint') as 'NonYielding_Tasks',        data.value('/system/@pageFaults')[1], 'bigint') as 'Page_Faults',        data.value('/system/@latchWarnings')[1], 'bigint') as 'Latch_Warnings',        data.value('/system/@BadPagesDetected')[1], 'bigint') as 'BadPages_Detected',        data.value('/system/@BadPagesFixed')[1], 'bigint') as 'BadPages_Fixed' from SpServerDiagnosticsResult where component_name like 'system' go  -- Resource Monitor select data.value('/Record/ResourceMonitor/Notification')[1], 'VARCHAR(max)') AS [Notification],        data.value('/resource/memoryReport/entry[@description='Working Set']/@value)[1], 'bigint')/1024 AS [SQL_Mem_in_use_MB],        data.value('/resource/memoryReport/entry[@description='Available Paging File']/@value)[1], 'bigint')/1024 AS [Avail_Pagefile_MB], </pre>	

```

        data.value('/resource/memoryReport/entry[@description='Available Physical Memory']/@value)[1]',
        'bigint')/1024 AS [Avail_Physical_Mem_MB],
        data.value('/resource/memoryReport/entry[@description='Available Virtual Memory']/@value)[1]',
        'bigint')/1024 AS [Avail_VAS_MB],
        data.value('/resource/@lastNotification')[1], 'varchar(100)') as 'LastNotification',
        data.value('/resource/@outOfMemoryExceptions')[1], 'bigint') as 'OOM_Exceptions'
from SpServerDiagnosticsResult
where component_name like 'resource'
go

-- Nonpreemptive waits
select waits.evt.value('@waitType'), 'varchar(100)') as 'Wait_Type',
       waits.evt.value('@waits'), 'bigint') as 'Waits',
       waits.evt.value('@averageWaitTime'), 'bigint') as 'Avg_Wait_Time',
       waits.evt.value('@maxWaitTime'), 'bigint') as 'Max_Wait_Time'
from SpServerDiagnosticsResult
      CROSS APPLY data.nodes('/queryProcessing/topWaits/nonPreemptive/byDuration/wait') AS waits(evt)
where component_name like 'query_processing'
go

-- Preemptive waits
select waits.evt.value('@waitType'), 'varchar(100)') as 'Wait_Type',
       waits.evt.value('@waits'), 'bigint') as 'Waits',
       waits.evt.value('@averageWaitTime'), 'bigint') as 'Avg_Wait_Time',
       waits.evt.value('@maxWaitTime'), 'bigint') as 'Max_Wait_Time'
from SpServerDiagnosticsResult
      CROSS APPLY data.nodes('/queryProcessing/topWaits/preemptive/byDuration/wait') AS waits(evt)
where component_name like 'query_processing'
go

-- CPU intensive requests
select cpureq.evt.value('@sessionId'), 'bigint') as 'SessionID',
       cpureq.evt.value('@command'), 'varchar(100)') as 'Command',
       cpureq.evt.value('@cpuUtilization'), 'bigint') as 'CPU_Utilization',
       cpureq.evt.value('@cpuTimeMs'), 'bigint') as 'CPU_Time_ms'
from SpServerDiagnosticsResult
      CROSS APPLY data.nodes('/queryProcessing/cpuIntensiveRequests/request') AS cpureq(evt)
where component_name like 'query_processing'
go

-- Blocked Process Report
select blk.evt.query('.') as 'Blocked_Process_Report_XML'
from SpServerDiagnosticsResult
      CROSS APPLY data.nodes('/queryProcessing/blockingTasks/blocked-process-report') AS blk(evt)
where component_name like 'query_processing'
go

-- IO
select data.value('/ioSubsystem/@ioLatchTimeouts')[1], 'bigint') as 'Latch_Timeouts',
       data.value('/ioSubsystem/@totalLongIos')[1], 'bigint') as 'Total_Long_IOs'
from SpServerDiagnosticsResult
where component_name like 'io_subsystem'
go

-- Event information
select xevts.evt.value('@name'), 'varchar(100)') as 'xEvent_Name',
       xevts.evt.value('@package'), 'varchar(100)') as 'Package',
       xevts.evt.value('@timestamp'), 'datetime') as 'xEvent_Time',
       xevts.evt.query('.') as 'Event Data'
from SpServerDiagnosticsResult
      CROSS APPLY data.nodes('/events/session/RingBufferTarget/event') AS xevts(evt)
where component_name like 'events'
go

```

## Siehe auch

[Failoverrichtlinie für Failoverclusterinstanzen](#)