

Handling dropped connections using TCP Keepalive

Last updated by | Holger Linke | Mar 1, 2023 at 4:40 AM PST

Contents

- [Why & What is TCP Keep-alive](#)
- [How it works](#)
- [Configuring TCP Keep-alive \(Windows\)](#)
- [Keep-alive parameters](#)
- [Enable from Application](#)
 - [jTDS JDBC driver](#)
 - [Confirm keepalive working](#)
- [Side effects](#)
- [Recommendation](#)
- [More Information - jTDS](#)
- [Related links](#)

Why & What is TCP Keep-alive

With TCP Keepalive, you will be able to check your connected socket (also known as TCP sockets), and determine whether the connection is still up and running or if it has broken. Without a keepalive signal, intermediate NAT-enabled routers can drop the connection after timeout.

- Keep-alive packet is simply an ACK with the sequence number set to one less than the current sequence number for the connection. A server receiving one of these ACKs from client responds with an ACK for the current sequence number. Keep-alives can be used to verify that the server/endpoint at the remote end of a connection is still available. TCP keep-alives can be sent once every `KeepAliveTime` (*default is 7,200seconds but configurable*) if no other data or higher-level keep-alives have been carried over the TCP connection.

How it works

- TCP keepalive starts with sending a TCP ack segment to the remote end. The ack segment has sequence number one less than from the current sequence number. If the receiver has an active connection, then ACK is returned to the sender and the probe stops.
- When the receiver host is not reachable or TCP not running on the destination port, there will be no response for a keepalive probe, the sender keeps repeating the probe after an interval (default value). After a fixed number of tries, the connection is marked as down. TCP clears the connection resources and indicates to the user for connection close.
- TCP keep-alives are disabled by default, but Windows Sockets applications can use the `setsockopt()` function to enable them.

Configuring TCP Keep-alive (Windows)

To avoid dropping idle connections by a network component, the following registry settings (or their non-Windows equivalents) should be set on the operating system where the driver is loaded.

Registry Setting	Recommended Value in Milliseconds
HKEY_LOCAL_MACHINE \ SYSTEM \ CurrentControlSet \ Services \ Tcpip \ Parameters \ KeepAliveTime	30000
HKEY_LOCAL_MACHINE \ SYSTEM \ CurrentControlSet \ Services \ Tcpip \ Parameters \ KeepAliveInterval	1000
HKEY_LOCAL_MACHINE \ SYSTEM \ CurrentControlSet \ Services \ Tcpip \ Parameters \ TcpMaxDataRetransmissions	10

Restart the computer for the registry settings to take effect.

These settings will have the effect of disconnecting an unresponsive connection within 10 to 40 seconds. After a keep alive packet is sent, if no response is received, it will be retried every second up to 10 times. If no response is received during that time, the client-side socket is disconnected. Depending on your environment, you may want to increase the KeepAliveInterval to accommodate known disruptions (like virtual machine migrations) that might cause a server to be unresponsive for longer than 10 seconds

Keep-alive parameters

- **Keepalive time** is the duration between two keepalive transmissions in idle condition. TCP keepalive period is required to be configurable and by default is set to no less than 2 hours. The parameter controls how often TCP attempts to verify that an idle connection is still intact by sending a keep-alive packet. If the remote system is still reachable and functioning, it acknowledges the keep-alive transmission. Keep-alive packets are not sent by default. This feature may be enabled on a connection by an application.
- **Keepalive interval** is the duration between two successive keepalive retransmissions, if acknowledgement to the previous keepalive transmission is not received. This parameter determines the interval between keep-alive retransmissions until a response is received. Once a response is received, the delay until the next keep-alive transmission is again controlled by the value of **KeepAlive Time**. The connection is aborted after the number of retransmissions specified by TcpMaxDataRetransmissions have gone unanswered.
- **Keepalive retry** is the number of retransmissions to be carried out before declaring that remote end is not available

Enable from Application

Even if TCP KeepaliveTime and TCPKeepAliveInterval registry keys are set to a specific value (TCPIP driver uses the default values even if we don't set these registry keys from the registry), TCPIP driver won't start sending TCP Keepalives until Keepalives are enabled via various methods at upper layers (layers above TCPIP driver). \

Native Socket applications can enable TCP keepalives by using anyone of the following methods:

- `setsockopt()` with `SO_KEEPALIVE` option
- `WSAIoctl()` with `SIO_KEEPALIVE_VALS` option (it's also possible to change Keepalive timers with this API call dynamically on a per-socket basis)

jTDS JDBC driver

Check the connection properties and ensure `socketKeepAlive` (default - false) is set to 'true' to enable TCP/IP keep-alive messages.

Make sure to review the "More Information" section below for concerns regarding jTDS.


Confirm keepalive working

For connections successfully set up with keep alive, we will see the keep alive timer available for each connection.


Side effects

- Generates extra traffic in order to ensure the connection is alive.
- This extra hop can have an impact on routers and firewalls.

Recommendation





- To ensure stable, reliable, and secure connectivity, we strongly recommend user(s) to update (or contact the application provider) to the latest version of a supported driver like [Microsoft JDBC Driver for SQL Server](#) .
- This driver is available at no additional charge and provides Java database connectivity from any Java application, application server, or Java-enabled applet.
- This driver is a Type 4 JDBC driver that provides database connectivity through the standard JDBC application program interfaces (APIs).

More Information - jTDS

jTDS is an open source Java (type 4) JDBC 3.0 driver for Microsoft SQL Server 6.5, 7, 2000, 2005, 2008 and 2012. The [latest version](#)  is from June 2013; it has no concept of cloud databases, proxy vs. redirect, retry logic, network packet handling over intermediate proxies and gateways, or any feature that had been introduced after SQL Server 2012.

jTDS is known to have issues when sending and receiving TDS packages that are spanning several network packages. It leads to dropped connections that cannot be worked around. See related article [Error 4014](#) for further information about symptoms and mitigation steps.

Related links

- [TCP 3 way handshake - basics](#) 
- [TCP connectivity handling](#) 
- [Microsoft JDBC Driver for SQL Server](#) 
- [Use Java and JDBC with Azure SQL Database](#) 

How good have you found this content?

