java.net.SocketException: Connection reset

Asked 16 years, 3 months ago Modified 8 months ago Viewed 1.3m times

189

I am getting the following error trying to read from a socket. I'm doing a `readInt()` on that `InputStream`, and I am getting this error. Perusing the documentation suggests that the client part of the connection closed the connection. In this scenario, I am the server.

I have access to the client log files and it is not closing the connection, and its log files suggest I am closing the connection. So does anybody have an idea why this is happening? What else to check for? Does this arise when there are local resources that are perhaps reaching thresholds?

I do note that I have the following line:

```
socket.setSoTimeout(10000);
```

just prior to the `readInt()`. There is a reason for this (long story), but just curious, are there circumstances under which this might lead to the indicated error? I have the server running in my IDE, and I happened to leave my IDE stuck on a breakpoint, and I then noticed the exact same errors begin appearing in my own logs in my IDE.

Anyway, just mentioning it, hopefully not a red herring.

java sockets network-programming connection socketexception

Share
Improve this question
Follow

edited May 12, 2023 at 0:47
Kirby
15.8k ● 11 ● 95 ● 108

asked Sep 15, 2008 at 13:36
Darryl

2 Do you have stack traces from both sides? Can you describe the network architecture a bit more? (Over the wild Internet? On the same machine? Somewhere in between?) Does it happen all the time? Or intermittently? – Stu Thompson Sep 15, 2008 at 13:45

See stackoverflow.com/questions/585599/… – Raedwald Nov 13, 2018 at 22:13

I've had the same problem using WAMP, but fixed it working on a remote server. – Florian-Martin Feb 2, 2021 at 12:32

Generally, I am asking, can a SQL password expire result in this problem? – Sm Srikanth Mar 10, 2022 at 7:20

17 Answers

Sorted by: Highest score (default)

148

There are several possible causes.

1. The other end has deliberately reset the connection, in a way which I will not document here. It is rare, and generally incorrect, for application software to do this, but it is not unknown for commercial software.
2. More commonly, it is caused by writing to a connection that the other end has already closed normally. In other words an application protocol error.
3. It can also be caused by closing a socket when there is unread data in the socket receive buffer.
4. In Windows, 'software caused connection abort', which is not the same as 'connection reset', is caused by network problems sending from your end. There's a Microsoft knowledge base article about this.
5. Windows, but not Unix, Linux etc., resets a connection if a process exists without closing a socket. The other OSes close it properly.

Share

Improve this answer

Follow

edited Mar 29 at 15:59

answered Nov 29, 2010 at 4:11

user207421

311k ● 44 ● 320 ● 488

@MattLyons Thanks. There are much better MSDN articles than that. Frankly I find that one hard to believe. A connection won't even exist until the correct source and target IP addresses have been established. The MSDN articles I have seen refer to persistent network errors timing out the connection. – user207421 May 30, 2012 at 10:13

1 Given your first 3 points, this would be interesting for you to see a decade later – zanderwar Mar 24, 2021 at 4:51

1 @zanderwar Thanks. Plagiarism abounding over there. – user207421 Mar 24, 2021 at 23:38

This answer appears here too net-informations.com/java/err/reset.htm – Alan Telles Sep 20, 2022 at 12:09

@AlanTelles And as usual when plagiarists paraphrase they introduce errors. I didn't say anything about the port not being open, which would not cause this exception; and the part about the socket being closed is wrong as well. – user207421 Apr 4, 2023 at 10:00

72

Connection reset simply means that a TCP RST was received. This happens when your peer receives data that it can't process, and there can be various reasons for that.

The simplest is when you close the socket, and then write more data on the output stream. By closing the socket, you told your peer that you are done talking, and it can forget about your connection. When you send more data on that stream anyway, the peer rejects it with an RST to let you know it isn't listening.

In other cases, an intervening firewall or even the remote host itself might "forget" about your TCP connection. This could happen if you don't send any data for a long time (2 hours is a common time-out), or because the peer was rebooted and lost its information about active connections. Sending data on one of these defunct connections will cause a RST too.

Update in response to additional information:

Take a close look at your handling of the `SocketTimeoutException`. This exception is raised if the configured timeout is exceeded while blocked on a socket operation. The state of the socket itself is not changed when this exception is thrown, but if your exception handler closes the socket, and then tries to write to it, you'll be in a connection reset condition. `setSoTimeout()` is meant to give you a clean way to break out of a `read()` operation that might otherwise block forever, without doing dirty things like closing the socket from another thread.

Share

Improve this answer

Follow

edited Oct 8, 2014 at 16:20

answered Sep 15, 2008 at 14:03

erickson

269k ● 59 ● 401 ● 497

Not correct in several respects. Garbage collection and process exits both cause proper closes, not resets, but a close followed by a write by the peer can induce a reset rather than an EOS. SocketTimeoutExceptions are only raised if the reader has set a read timeout. – user207421 Sep 30, 2012 at 0:21

And it doesn't always mean an RST was received. It can also mean one was generated by this side. – user207421 Apr 14, 2017 at 17:35

And you can't write to a socket that you have already closed. You will get a `SocketException: socket closed`, and the peer will not get an RST. Answer is completely incorrect. – user207421 Nov 12, 2020 at 6:46

Whenever I have had odd issues like this, I usually sit down with a tool like WireShark and look at the raw data being passed back and forth. You might be surprised where things are being disconnected, and you are only being *notified* when you try and read.

22

Share Improve this answer Follow

answered Sep 15, 2008 at 13:45

GEOCHET
21.3k ● 15 ● 77 ● 99

16

You should inspect full trace very carefully,

I've a server socket application and fixed a `java.net.SocketException: Connection reset` case.

In my case it happens while reading from a clientSocket `Socket` object which is closed its connection because of some reason. (Network lost,firewall or application crash or intended close)

Actually I was re-establishing connection when I got an error while reading from this Socket object.

```
Socket clientSocket = ServerSocket.accept();
is = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
int readed = is.read(); // WHERE ERROR STARTS !!!
```

The interesting thing is `for my JAVA Socket` if a client connects to my `ServerSocket` and close its connection without sending anything `is.read()` is being called repeatedly.It seems because of being in an infinite while loop for reading from this socket you try to read from a closed connection. If you use something like below for read operation;

```
while(true)
{
  Receive();
}
```

Then you get a stackTrace something like below on and on

```
java.net.SocketException: Socket is closed
    at java.net.ServerSocket.accept(ServerSocket.java:494)
```

What I did is just closing ServerSocket and renewing my connection and waiting for further incoming client connections

```
String Receive() throws Exception
{
```

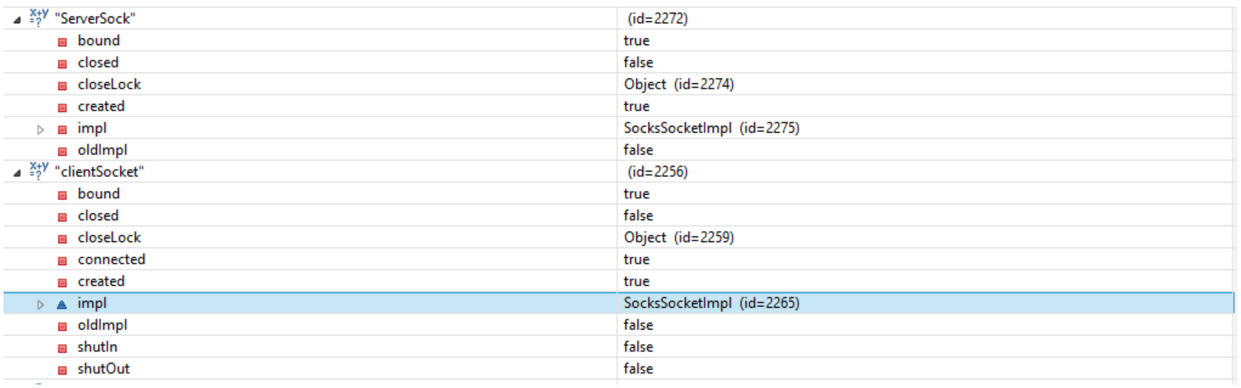
```
try {
            int readed = is.read();
           ....
}catch(Exception e)
{
        tryReConnect();
        logit(); //etc
}


//...
}
```

This reestablises my connection for unknown client socket losts

```
private void tryReConnect()
        {
            try
            {
                ServerSocket.close();
                //empty my old lost connection and let it get by garbage col.
immediately
                clientSocket=null;
                System.gc();
                //Wait a new client Socket connection and address this to my
local variable
                clientSocket= ServerSocket.accept(); // Waiting for another
Connection
                System.out.println("Connection established...");
            }catch (Exception e) {
                String message="ReConnect not successful "+e.getMessage();
                logit();//etc...
            }
        }
```

I couldn't find another way because as you see from below image you can't understand whether connection is lost or not without a `try and catch` ,because everything seems right . I got this snapshot while I was getting `Connection reset` continuously.



```
◢ "ServerSock"          (id=2272)
    bound               true
    closed              false
    closeLock           Object  (id=2274)
    created             true
  ▷ impl                SocksSocketImpl  (id=2275)
    oldImpl             false
◢ "clientSocket"        (id=2256)
    bound               true
    closed              false
    closeLock           Object  (id=2259)
    connected           true
    created             true
  ▷ impl                SocksSocketImpl  (id=2265)
    oldImpl             false
    shutIn              false
    shutOut             false
```

Share

Improve this answer

Follow

edited Nov 23, 2020 at 22:32

answered Jul 31, 2015 at 8:20

Davut Gürbüz
5,716 ● 4 ● 50 ● 86

@MarquisofLorne yes correct, that's why that routine is in another while loop. – Davut Gürbüz Nov 12, 2020 at 7:48

@MarquisofLorne I noticed after reading my own answer it needed to be 'is being called repeatedly'. Thanks for pointing out. 5y ago :). I improved it now actually, based on reading -1, timeout exception etc. added. Ungraceful connection drops are still problem, there are different inactivity management methods people follow. – Davut Gürbüz Nov 23, 2020 at 22:32

Why did you close the server socket? None of this makes sense. – user207421 Nov 23, 2023 at 19:22

@user207421 to give it a fresh start. In my setting there are so many network layers edge to edge and sometimes while one end see `ESTABLISHED` , other side observe `SYNC_SENT` .. in such cases we restart the entire app. Also if the thread is blocked while `accept` it's caught and retry routine works again with a fresh start.. that's what I remember from 7y back. Thanks for your kind comment. – Davut Gürbüz Nov 24, 2023 at 9:56

12

Embarrassing to say it, but when I had this problem, it was simply a mistake that I was closing the connection before I read all the data. In cases with small strings being returned, it worked, but that was probably due to the whole response was buffered, before I closed it.

In cases of longer amounts of text being returned, the exception was thrown, since more then a buffer was coming back.

You might check for this oversight. Remember opening a URL is like a file, be sure to close it (release the connection) once it has been fully read.

Share Improve this answer Follow

answered Jul 23, 2013 at 17:10

Scott S
501 ● 8 ● 20

I had this problem. Turns out the webClient was being overwhelmed as it was initialised early in the program and never reset. Re-initialising it at the point of each call solved it for me. – Chidozie Nnachor Mar 7, 2023 at 13:00

7

I had the same error. I found the solution for problem now. The problem was client program was finishing before server read the streams.

Share Improve this answer Follow

answered Mar 27, 2011 at 20:30

kml_ckr
2,241 ● 3 ● 22 ● 36

That would not cause this exception on its own. – user207421 Jun 12, 2012 at 21:20

it would if System.exit(0) kills the program and no one calls socket.close() as the connection is in a bad state and was not closed properly. sooo more properly said he had a client program that shutdown without closing sockets ;) which is a bad thing and should be fixed.
– Dean Hiller Nov 9, 2012 at 20:29

1 @DeanHiller No it wouldn't. The operating system would close the socket the same way the application should have. – user207421 Oct 5, 2013 at 23:27

@EJP ....I am not sure...I just know we could reproduce it with System.exit but I don't remember the OS/config as that was quite some time ago....calling socket.close() on the server prevented connection reset and it behaved more properly. – Dean Hiller Oct 7, 2013 at 12:14

2 @DeanHiller The *reading* process exited before the writing process had finished writing.
– user207421 May 18, 2014 at 11:57

6

I had this problem with a SOA system written in Java. I was running both the client and the server on different physical machines and they worked fine for a long time, then those nasty connection resets appeared in the client log and there wasn't anything strange in the server log. Restarting both client and server didn't solve the problem. Finally we discovered that the heap on the server side was rather full so we increased the memory available to the JVM: problem solved! Note that there was no OutOfMemoryError in the log: memory was just scarce, not exhausted.

Share
Improve this answer
Follow

edited Sep 14, 2017 at 17:18

answered May 8, 2015 at 12:24
Pino
9,233 ● 7 ● 57 ● 76

1 There is nothing about `OutOfMemoryError` in the question. – user207421 Mar 24, 2021 at 4:09

1 Yes, I know, I didn't say otherwise. – Pino Mar 25, 2021 at 13:46

3

Check your server's Java version. Happened to me because my Weblogic 10.3.6 was on JDK 1.7.0_75 which was on TLSv1. The rest endpoint I was trying to consume was shutting down anything below TLSv1.2.

By default Weblogic was trying to negotiate the strongest shared protocol. See details here: Issues with setting https.protocols System Property for HTTPS connections.

I added verbose SSL logging to identify the supported TLS. This indicated TLSv1 was being used for the handshake.
`-Djavax.net.debug=ssl:handshake:verbose:keymanager:trustmanager -Djava.security.debug=access:stack`

I resolved this by pushing the feature out to our JDK8-compatible product, JDK8 defaults to TLSv1.2. For those restricted to JDK7, I also successfully tested a workaround for Java 7 by upgrading to TLSv1.2. I used this answer: How to enable TLS 1.2 in Java 7

Share Improve this answer Follow

answered Aug 15, 2019 at 20:13
Sumiya
337 ● 4 ● 5

many thanks, you show me a new direction to fix my issue. and finally I found it is the case!!!
– karl li Jun 1, 2020 at 7:39

SSL protocol mismatches do not cause connection resets. They cause handshake failures.
– user207421 Nov 12, 2020 at 6:48

3

I was also getting HTTP 500 error "java.net.SocketException: Connection reset", and after couple of days of analyze it turned out that problem is caused by AWS NAT Gateway. I hope it will help someone to save a time and resolve the problem.

Infrastructure:
AWS API Gateway (HTTP), AWS CloudMap, AWS Fargate.

Problem investigation:
After deep investigation it turned out that the problem is caused by AWS NAT Gateway which keeps information about specific NAT translations for 350 sec and removes it if there is no traffic in that period. My application uses Java HTTP client, which keeps connections for 20 minutes. Due to this, first 1 or 2 requests after time of 350 seconds ends with HTTP 500 Connection Reset. NAT Gateway doesn't have information about that specific translations and it responds with RST flag - that's why I saw "Connection Reset" in logs.

Implemented solution:
Changing HTTP Java client to HTTP Apache client. HTTP Apache client properly keeps connection alive. Optionally you can adjust native Java HTTP client to send TCP keep-alive, or to close inactive connection earlier than 350sec.

Share Improve this answer Follow

answered Jul 3, 2023 at 8:38
leksykoon
41 ● 4

1

I also had this problem with a Java program trying to send a command on a server via SSH. The problem was with the machine executing the Java code. It didn't have the permission to connect to the remote server. The write() method was doing alright, but the read() method was throwing a java.net.SocketException: Connection reset. I fixed this problem with adding the client SSH key to the remote server known keys.

Share Improve this answer Follow

answered Jul 10, 2014 at 16:07
pmartin8
1,584 ● 1 ● 20 ● 37

0

In my experience, I often encounter the following situations;

1. If you work in a corporate company, contact the network and security team. Because in requests made to external services, it may be necessary to **give permission for the relevant endpoint.**
2. Another issue is that the **SSL certificate may have expired** on the server where your application is running.

Share Improve this answer Follow

answered Jun 5, 2020 at 14:00
bmck
229 ● 2 ● 7

6 SSL certificates do not cause connection resets. – user207421 Nov 12, 2020 at 6:49

0

In my case was `DNS problem` .
I put in `host file` the resolved IP and everything works fine. Of course it is not a permanent solution put this give me time to fix the DNS problem.

Share Improve this answer Follow

answered Jul 22, 2021 at 7:18
tsotzolas
397 ● 1 ● 11

This does not cause connection resets. – user207421 Nov 23, 2023 at 19:23

0

I've seen this problem. In my case, there was an error caused by reusing the same ClientRequest object in an specific Java class. That project was using Jboss Resteasy.

1. Initially only one method was using/invoking the object ClientRequest (placed as global variable in the class) to do a request in an specific URL.
2. After that, another method was created to get data with another URL, reusing the same ClientRequest object, though.

The solution: in the same class was created another ClientRequest object and exclusively to not be reused.

Share Improve this answer Follow

answered Jul 22, 2021 at 16:18
Thiago Ferreira
190 ● 2 ● 6

0

In my case it was problem with TSL version. I was using Retrofit with OkHttp client and after update ALB on server side I should have to delete my config with **connectionSpecs**:

```
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
        List<ConnectionSpec> connectionSpecs = new ArrayList<>();
        connectionSpecs.add(ConnectionSpec.COMPATIBLE_TLS);
  //      clientBuilder.connectionSpecs(connectionSpecs);
```

So try to remove or add this config to use different TSL configurations.

Share Improve this answer Follow

answered Feb 22, 2022 at 19:39
Djek-Grif
1,496 ● 19 ● 18

0

I used to get the 'NotifyUtil::java.net.SocketException: Connection reset at java.net.SocketInputStream.read(SocketInputStream.java:...' message in the Apache Console of my Netbeans7.4 setup.

I tried many solutions to get away from it, what worked for me is enabling the TLS on Tomcat.

Here is how to:

> Create a keystore file to store the server's private key and self-signed certificate by executing the following command:

> Windows:
>
> "%JAVA_HOME%\bin\keytool" -genkey -alias tomcat -keyalg RSA
>
> Unix:
>
> $JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA
>
> and specify a password value of "changeit".

As per https://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html (This will create a .keystore file in your localuser dir)

Then edit server.xml (uncomment and edit relevant lines) file (%CATALINA_HOME%apache-tomcat-7.0.41.0_base\conf\server.xml) to enable SSL and TLS protocol:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
               maxThreads="150" scheme="https" secure="true"
               clientAuth="false" sslProtocol="TLS" keystorePass="changeit" />
```

I hope this helps

Share Improve this answer Follow

answered Sep 23, 2022 at 9:11
MaVRoSCy
17.8k ● 15 ● 84 ● 128

0

Overall, to find the root cause of the issue, I've found helpful to trace incoming/outgoing tcp packets with a tool such as tcpdump. I've found this article useful to show through a real session the detection of the root cause of the Connection Reset Issue by using a sample reproducer.

Share Improve this answer Follow

answered Nov 23, 2023 at 19:12
Francesco Marchioni
4,328 ● 2 ● 29 ● 44

-1

I've encountered a similar socket exception. We've setup an MFT SFTP connection with two sequential control-m jobs each sending 10 files of 1Mo each for 1 minute to an external server. Sometimes, a file was missing on the external server with the following error in our MFT logs:

```
Error writing data to final destination. | Proxy error in reading file.
(...)
```

```
Failed to connect to ssh server. Exception: java.net.SocketTimeoutException
```

We fixed this anomaly by adding a 2 minutes delay between the scheduling of the two control-m jobs.

Share

Improve this answer

Follow

edited Nov 27, 2023 at 13:33

answered Nov 3, 2023 at 15:18

Michael Fayad

1,316 ● 1 ● 18 ● 38

A socket timeout is not a connection reset. – user207421 Nov 23, 2023 at 19:24

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.