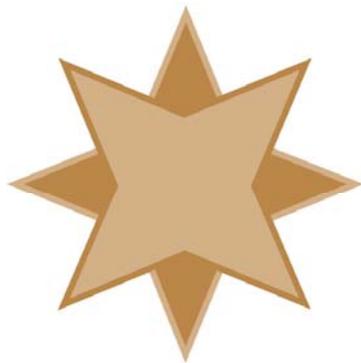# Status 95/3111 Discussion and FAQ List

A White Paper From

www.GoldstarSoftware.com

For more information, see our web site at
http://www.goldstarsoftware.com

Status 95/3111 Discussion and FAQ List
Assembled by Goldstar Software Inc. - Last Updated: 09/18/2000

The dreaded "Status 95" -- a common problem for many sites, never seen at others. Why is this issue so hard to track down and resolve? Whose problem is a status 95? Why can't the software developer fix it? These are just some of the common questions relating to the Btrieve Status 95 and Pervasive.SQL Status 3111. This discussion and FAQ was assembled to help address the wide variety of questions created when this error occurs in an application.

The first group of *Just Enough Information* contains some basic points of understanding on where the status 95's come from and how to usually avoid them. The second group *(Too Much Information)* describes in detail the session maintenance issues in SPX and TCP, and also discusses the problems with Novell's SPX algorithm. Finally, the last section contains *Resolution Information* which contains a few tips to use if you are suffering from chronic Status 95 problems.

Just Enough Information
Here are some overview points about the status 95 issues.

How Does Pervasive Software Define a Status 95?
The Pervasive manual describes a status 95 as simply "The session is no longer valid." However, the CodeHelp Windows Help file sheds some more light on this issue:

95: The session is no longer valid.
The MicroKernel returns this status code for one of the following reasons:

- *The previously established session is no longer active due to an error at the workstation, at the server, or on the network. Verify that the workstation is still attached to the server, and then reload the Btrieve Requester.*
- *The server MicroKernel has reached the maximum number of SPX sessions. Use the Monitor utility to check this statistic. Use the Setup utility to specify a higher value for the Number of Sessions configuration option.*
- *This status code could indicate a time delay problem if the workstation does not receive a response back from the server in an appropriate time frame or after an appropriate number of retries. Refer to your network configuration documentation for information about increasing timeout and retry parameters. This is often necessary in a WAN environment or a LAN configuration with heavy network traffic.*
- *For NetWare servers, verify that the Maximum Packet Receive Size configuration option is appropriate for you environment. For example, the setting should be 1500 for an Ethernet LAN or 4096 for a Token-Ring LAN.*
- *For NetWare servers, ensure that the SPX timeout parameters are set as follows in both the workstation's NET.CFG file and the server's SPXCONFG.NLM file:*

  SPX VERIFY TIMEOUT=54
  SPX LISTEN TIMEOUT=108
  SPX ABORT TIMEOUT=540

  These three values must have a 1:2:10 ratio. You can increase these values to at most three times the default. If you continue to receive this status code after increasing these values, the problem is most likely not related to these settings.

- *For Windows NT servers, verify that the Maximum Packet Size registry setting is 576 decimal or 240h. The path to the MaxPktSize registry setting is: HKEY_LOCAL_MACHINE\System\currentControlSet\Services\NwlnkIPX\NetConfig\MaxPktSize.*

If you continue to receive this status code after increasing the network timeout parameters, this status code usually indicates a problem with network communications. Verify that you have current network cards and drivers; for

example, incompatible LAN card drivers can also cause this status code to occur. Consult you LAN administrator for network communication troubleshooting.

What Does This Really Mean To Me?
In short, this means that the low-level connection between the database engine on the file server and the remote requester on the workstation has broken for some reason.

I Contacted the Software Developer, But They Can't Help Me!
This problem is **not** an application development problem, except for the fact that they chose Btrieve as their development platform, which, in my own personal opinion, makes for the best platform for most applications with a high rate of transactional database operations, such as accounting packages or data tracking applications. Btrieve, along with other SPX applications like Novell's RConsole and TCP applications such as FTP, rides along on the lower level transport protocol, which is supposed to guarantee delivery and sequencing of messages correctly. If, due to bugs in the protocols or problems in the physical network, the connection is dropped, then a Status 95 will result. There is nothing that an application developer can do, no matter how much they want to, which can get around this.

Can I get Status 95 errors with a local (workstation) engine?
By definition, No. The local engine runs on the individual workstations, and downloads ALL data required for operation. Therefore, there is no networking session from the application to the database engine, and this can never be lost. However, similar networking problems CAN appear as Status 2 errors to the local engine which are not reproducable if you retry the operation.

I get a Status 95 at exactly the same spot in the application every time. Why?
This type of Status 95, one that occurs at exactly the same location in the application is still a communications problem, but it is not the same issue as discussed above. In this case, which is usually found with 32-bit Windows applications and NetWare servers, the application has tried to write a record which is larger than approximately 480 bytes. When a DOS or Windows 16-bit application makes a request like this, the record is broken down into small, 512-byte chunks and sent in sequence to the server using multiple packets. However, the 32-bit Windows requester understands how to communicate with larger packets, so it sends the data in a single packet.

If the Btrieve communications component on the NetWare server (BSPXCOM) has not been configured with a -r parameter to specify a larger packet size, then it will only understand the 512-byte packets, and it ignores the larger packets. Since the server never responds that it received the large packet, the workstation retries the request again, looping on the retry until it times out, usually after 10 retries.

Note that this will be seen at exactly the same point in the application, and it will occur every time that code is executed.

The solution is simple -- add a -r=1500 parameter to the BSPXCOM load line in BSTART.NCF for Ethernet networks, or use -r=4096 for Token Ring networks. Restart the engine to load the new parameter, and the problem should disappear.

My network people say that my network is fine!
That's what they all say! Actually, there are 7 several layers to the OSI model, upon which most networking systems are based. Each level provides some basic services for higher levels to leverage in providing more functionality. Most of what users see as a "network" is the highest, or application layer, upon which protocols such as NCP (NetWare Core Protocol) and SMB (Server Message Block) are based. These protocols are used for basic file and print services, but all reside at layer 7, which incorporates many levels or error checking, retry code, and other recovery mechanisms to prevent lost packets. Since SPX and TCP reside at layer 4, there are fewer recovery mechanisms available in case something breaks.

Often, a "bad" network may exist on which no problems are evident. Users may see slow response time, but they can easily attribute those delays to slow servers or congested network lines. In actuality, the slowness may be caused by the inherent retry mechanisms in the higher level protocols. The only time they actually see the problem appear is when a lower-level transport protocol is required.

Problems along this line are usually caused by poor or out-of-spec cabling, out-of-spec networking equipment (i.e. too many hubs), bad cable termination, excessive collisions on Ethernet, or even congested routers. These issues can usually be overcome by retrying the transmissions, but performance penalties result.

There are a few ways to identify these lower-level problems.

- The easiest way is to quickly look at the cabling in the wiring closet of connected to the hubs. If the cables are well-made, then the insulation will be crimped inside the connector, providing a good, solid connection and maintaining wire twists right to the end of the wire. If a length of bare strands are visible outside the connector, then the termination likely needs improvement.
- A simple 8-wire cable tester can verify that the wires are indeed wired straight-through, but this is not the only requirement for higher speed communications. Just as important is the fact that the wires must be properly twisted -- wires should be color-paired as 1&2, 3&6, 4&5, and 7&8, with the odd pin numbers being the White/color wire. The only easy way to verify this is with a high-end cable tester, otherwise known as a time domain reflectometer. Expect to pay a few thousand dollars for a good one. Better yet, have all of the cabling certified by a knowledgeable cable installer.
- Check the statistics and monitoring capabilities of the equipment you have. If there is a collision light on the hub which is continuously on, there may be a problem. If a hub or switch contains monitoring software, check the error counts on each port.
- Verify the duplex settings on every device. A recent on-site visit to troubleshoot continuing Status 95 errors found that the primary switch was set to full duplex, an interim media converter (fiber to UTP) was set to full duplex, and the destination hub fiber uplink port was set to half duplex. Resetting each component and enabling full duplex down the entire link resolved the problem completely!
- Obtain a high-end network analyzer, such as a Network Associates Sniffer, which can be used to analyze network traffic looking for retransmissions and other related problems which may not be visible at the lower levels. If you are strapped for resources, but you know someone with a Novell LANalyzer who can help you, then you may try to download Novell's LANalyzer for Windows Limited Functionality release. This was, at one time, freely available from Novell's web site as LZFWLF.EXE. It does not perform any protocol decoding and several functions are disabled, but it DOES allow you to capture packets and save them to a file for analysis by someone with a full LANalyzer.

Too Much Information

This section contains some of the details for the experienced networking staff who you might be blaming for the problem. It is designed as more of a layman discussion, and it is NOT intended to be a completely accurate representation. Anyone who wants more info can certainly check out the appropriate documentation.

How are sessions established?
When the requester (on the workstation) receives a request to open a remote data file, the first thing it does is to resolve the target name (i.e. F:\DATA\CUSTOMER.MKD) to a fully qualified name, which includes the server and volume or share. In our example, assuming that drive F is mapped to NWSERVER\VOL1:APPS it may resolve to the file NWSERVER\VOL1: APPS\DATA\CUSTOMER.MKD. On Windows NT Servers, this original location may be a sharename which is used natively, since shares can be seen by the server as well. When this step is complete, the requester attempts to determine the network address of the server and Btrieve engine, by checking the network for the appropriate service. If this fails, then we get a Status 20.

Once it has an address, which may be either an SPX address or a TCP address, the requester tries to contact the Btrieve engine at that address. If this fails, we get a Status 91. The steps for SPX session establishment are:

Source	Dest.	Packet Description
WS	Server	RIP Request to locate the proper SPX network number
Server	WS	RIP Response indicating correct path to requested SPX network
WS	Server	SPX Packet to establish connection (From open socket on workstation to socket 65535 on the server)

Server	WS	SPX: Server responds to the WS socket, provides a open socket number on the server side
WS	Server	SPX: Workstation acknowledges the server's socket and sequence numbers and prepares to start the session
Server	WS	SPX: Server acknowledges WS's information & says "go ahead"

After this exchange, the workstation sends its first request, usually either a Version call or Btrieve File Open call.

In the TCP/IP world, this exchange is just about identical.

Source	Dest.	Packet Description
WS	Server	ARP Request to locate the TCP/IP Address
Server	WS	ARP Response indicating correct path/node to the TCP/IP Address
WS	Server	TCP SYN packet to the well-known port 3351.
Server	WS	TCP ACK SYN packet acknowledging the request and synchronizing the TCP session at the server side.
WS	Server	TCP ACK packet telling the server that we're synchronized.

Interestingly, there is no wait for a reply from the server to re-verify the connection, and the workstation immediately starts sending data with his first request.

How are sessions maintained?
A normal SPX Session consists of 4 packets as shown below.

Source	Dest.	Packet Description
WS	Server	SPX: Btrieve operation request from the workstation.
Server	WS	SPX ACK: Server acknowledges the receipt of the request.
Server	WS	SPX: Server replies with the results of the operation.
WS	Server	SPX ACK: Workstation acknowledges the receipt of the reply.

Note that the workstation sends a request, then waits for the ACK from the server. The server sends the ACK, then follows up with the reply as soon as it can, and the workstation must ACK the reply before sending the next request. This bi-directional communications continues for the life of the connection. For local area networks, this is not really a problem. However, for WANs, the extra 64-byte ACK packets can increase the total time for an operation to complete. Additionally, when networks use large packet sizes (e.g. Gigabit Ethernet?), the 64-byte packet is padded to the minimum frame length by the network, further wasting bandwidth.

In the TCP world, sessions can be more efficient, since the separate ACK packets are not required. In fact, for a long-running conversation, one which makes a new request or reply right away every time, there are only two packets in each exchange, one with the request, and another with the reply.

How are large packets transmitted?
The above, ping-pong operation takes place for normal-sized packets. However, what happens when the data block is too large to be sent in a single network packet? Let's look at a request for data which takes two packets to fulfill, such as reading a large record.

Source	Dest.	Packet Description
WS	Server	SPX: Btrieve operation request from the workstation.
Server	WS	SPX ACK: Server acknowledges the receipt of the request.
Server	WS	SPX: Server replies with the first half of the results of the operation.

WS	Server	SPX ACK: Workstation acknowledges the receipt of the reply.
Server	WS	SPX: Server replies with the second half of the results of the operation.
WS	Server	SPX ACK: Workstation acknowledges the receipt of the reply.

Note that in this exchange, instead of seeing each side always transmitting twice in a row, we see alternating transmissions. In packet 3, which is a partial Reply, the server is only able to get 1/2 of the packet on the wire, so it must wait for an ACK from the workstation before sending the second half. (Note that this incurs a complete round-trip delay time on the network -- a real performance killer over wide area networks.) When it gets the ACK, it can send the next block of data, and so on until the total data block has been transmitted.

The same thing occurs when the request must be broken into several pieces. This is commonly done with a Btrieve Insert or Update request on a large record. Some applications have been seen to have records in the neighborhood of 20K, which can incur 40 partial request packets, meaning also 40 round-trip delays!

What is a Lost Packet?
What is a lost packet? This is a network packet, being sent from one computer to another, which doesn't make it to its destination. This can happen for several reasons:

1. Late Collisions occur when a network is out of spec, usually because of excessive line length. A collision occurs when two Ethernet nodes transmit on the same wire at the same time. The collision mangles the packet on the wire, so some end nodes do not see the packet at all. The lower level protocols (the physical and/or datalink layers) should detect a collision and retransmit the frame. However, network cards only look for a collision for a certain amount of time, as defined by the Ethernet specification. When the network is longer than it should be, it is possible to have a collision occur after the NIC stops watching for it, and the collision never gets detected, so the frame never gets retransmitted.
2. Bad cabling or electrical interference on substandard cabling can cause frames on the physical wire to be corrupted. Again, this may never get detected.
3. Overloaded routers or switches can be a problem. Routers can only handle a certain number of frames, based on line speed, available memory and CPU speed, among other factors. If a router is overloaded, it simply drops the excess frames, knowing that the higher-level protocols will retransmit the frame later on.
4. Bad hardware can also be a culprit. We've seen, in the real world, routers and switches which mangle the very packets they are burdened with forwarding. While this doesn't necessarily lose the packet, it can damage it beyond recognition by the receiving device.

Typically, lost packets are OK, because higher level protocols will retransmit to compensate.

What Happens if an SPX Packet is Lost?
Let's look at the four SPX packets and see what happens if any is lost.

Request Packet Lost
What happens if a request packet is lost? The workstation, upon sending the request, starts a timer which waits for the ACK to come back. If the ACK hasn't come in by the time the timer expires, then he assumes that the request was lost, and retransits the request. The server gets the request for the first time, sends the ACK, sends the reply, then waits for an ACK.

Request-ACK Packet Lost
If the request makes it to the server, but the ACK doesn't, then one of two things will happen. First, the server transmits the ACK packet, followed shortly by the reply packet, then waits for an ACK back. If the Request-ACK gets lost, then the Reply packet should also serve as an appropriate ACK, and the workstation should send the Reply-ACK.

Reply Packet Lost
If the server's reply packet is dropped, then the workstation will never send an ACK, and the server enters the same retry code as for the request packet. When the timer expires, if the ACK is still not yet received, then the reply is retransmitted. **Reply-ACK Packet Lost** Just like in the above, a reply-ACK should be followed by the next

request, if any, which serves as an ACK as well. If there is no other request, then the server will retransmit the reply packet and wait for an ACK again.

Note that in all of these cases, there are retransmissions to help guarantee receipt of every message. This protocol should be foolproof, right?

The Novell SPX Bug - Overview
Unfortunately for a lot of users across the country, the Novell operating system has a critical bug in its SPX handling code when certain packets are lost from the network. This is shown by chronic status 95 problems at a site, especially when WAN links or routers are heavily used.

The problem appears when a partial Request-ACK packet (sent from the server) is dropped by the network. A partial Request, as discussed above, is generated when a large amount of data is being sent to the server and must be broken into several request packets. This commonly happens in Btrieve applications, but with few other apps. Each of these requests must be ACKed by the server before the next portion can be transmitted.

As discussed above, if a Request-ACK packet is lost, the following reply packet should also trigger the ACK, but this is NOT the case here. Why? Because the next packet is NOT being sent by the server, the *workstation* is responsible for sending the next packet!

So, what really happens? The workstation never gets the ACK, so he assumes that his partial request packet never made it to the server. He then retransmits his packet to try again. The server, since it already received the partial request with that number and properly acknowledged it, discards the request **without sending another ACK**. This is the fault in the algorithm! If the workstation is retransmitting a packet, then the server must re-ACK the packet, so that the workstation can continue.

Since the workstation never gets the ACK from the second attempt, he tries again. And again. And again. And again. This repeats until the retry limit is reached, and the session fails completely. There is no way to re-establish this session at all, and the application is unable to recover gracefully.

It is important to understand the distinction between the bug in Novell's SPX stack and the cause of the dropped packet. The underlying network, as mentioned above, is responsible for dropping the packet. If this happens periodically, as it might even on a perfect network, an occasional status 95 may result. However, if this is happening excessively (i.e. chronic status 95's), then the problems at the networking level MUST be resolved. The bug is simply responsible for failure to allow a proper session retry to occur, which results in a status 95 after only **one** lost packet. (Normally, it would take **many** lost packets to effect a status 95.)

The Novell SPX Bug - Novell's Response
This bug exists in several places. First, Native SPX, which is used by Btrieve on the NetWare server, is an assembler routine built into SERVER.EXE. Secondly, the SPXS.NLM contains the C code for the SPX II protocol, which also appears to contain the same problems.

Novell's response to the native SPX is akin to "Dream on". The problem exists here, but it is highly doubtful that Novell will fix it due to the difficulties in properly maintaining assembly language source code and avoiding new bugs.

Since the SPXS code is written in C, it is easier to maintain. They have tried a few times to resolve this with interim patches, but none have been a sure-thing to date. Of course, even SPXS is becoming difficult to get changes for, since most of Novell's development staff is working on their TCP/IP protocol stacks nowadays.

NEW INFORMATION: Novell has recently released a patch file called SPXACK.ZIP, which contains the linked README file. This patch offers some promise for native SPX users which may resolve the problem! Like any other patch file, use with caution until it becomes an official part of the operating system patch list.

EVEN NEWER INFORMATION: This patch is included in the patch files for NetWare 4.11 Service Pack 7a! If someone is able to duplicate the status 95 bug (the bug itself, not just a dropped connection) in a NetWare

Resolution Information

So, you're having status 95 problems, huh? Here's a few things which may help to either resolve the problem or isolate it.

1. First of all, make sure that the physical network is as clean and up to spec as possible. This will help to reduce failures and improve response time for the entire network. We've seen 1000-node networks with VERY poor cabling complaining of problems. Unfortunately, until the cabling is cleaned up, there is nothing more which can be done.
2. Check any managed hubs, switches or routers, looking for late collisions, corrupted packets, runts, or other problems. These usually point to physical problems as well which must be fixed.
3. Try running the SmartScout Communications Stress Test. This test sends increasingly larger data blocks to the server (which will be broken down into partial packets, of course) over and over. Usually, this will fail quickly if you are dropping network packets. Remember that an overloaded router may be time-dependant, so don't just test this off-hours -- run the test during peak loads as well.
4. If you have random status 95's, look for lost communications links -- be it someone moving cables, short power outages on the hubs, or anything of the like. These external events can cause you to look for problems elsewhere which may not exist.
5. If you are using Windows applications to a NetWare server, seriously consider upgrading from your current Btrieve version to Pervasive.SQL 7, which includes Btrieve 7. This engine supports the TCP protocol to NetWare servers, which can eliminate the Novell Bug from consideration. Often, this resolves most, if not all, of the status 95 issues, since the TCP protocol stack doesn't appear to have the same problems as the SPX stack. The only issue here would be for large sites which do not have TCP/IP stacks on the workstations yet. Even so, it can be rolled out piecemeal, with the most critical workstations migrated to TCP first.
6. Lastly, we can wait for Novell to fix it, or for Pervasive to work around it. The Novell fix is the best, since it is exacerbating the lost packet issue into a full blown connection problem. Patch your NetWare 4.11 server to SP7a or later. The workaround is another story. While this issue would not be easy to work around, and it is not in Pervasive's best interest to work around it since they have the TCP stack available as an alternative, it is something which increasing pressure from software developers may force some day. Personally, I believe that this will add an incredible amount of overhead and platform-specific code which should not be necessary. Pervasive did identify a separate (but related) issue with the WBTICOMM.DLL file for 6.15 which uses a retry value of 5 times instead of the operating system default. You can download this file to test it here.

If you are plagued by status 95 problems and have not been able to find a solution, contact Goldstar Software and let us work with you to verify the causes and formulate the best options!