http://www.sqlservercentral.com/articles/Administration/99299/
Printed 2013/07/16 06:28AM

Orphaned Distribution Transactions

By Shashank Srivastava, 2013/05/17

One day you are performing an investigation in SQL Server into a performance issue. You run *sp_who2* to look for some blocking issues on the server, and you find SPID = -2 as your main blocker. It's at the top of a blocking chain with a dozen other transactions behind it trying to acquire resources. Until it's killed, nothing will happen.

Seeing the main blocker as SPID -2 you would think of killing it like a normal SPID.

```
KILL -2
```

After firing the kill command you receive this error. Now you are puzzled that why you am not able to kill the SPID.

```
Msg 6101, Level 16, State 1, Line 1

Process ID -2 is not a valid process ID. Choose a number between 1 and 2048
```

You start thinking about how to kill this negative SPID as you have not seeing this error in the past. You think of performing various tricks, including restarting the SQL Server. Well, there is no need to restart the SQL Server. Follow the below steps to resolve this issue. The minimum permission required to run the statements below is the processadmin privilege.

First run this:

```
Select req_transactionUOW
 from master..syslockinfo
 where req_spid = -2
```

This will return a 32 digit UOW number like 'FE4A57F2-28C5-44F9-8416-B08760DFE7E9'. Use this UOW to kill the main blocker.

```
KILL 'FE4A57F2-28C5-44F9-8416-B08760DFE7E9'
```

Run sp_who2 again and you will probably see that the SPID has disappeared. Now the explanation part, which will help you understand the reason behind the issue.

This negative SPID is known as a distributed transaction SPID or an orphaned distributed transaction SPID. What has happened is that when a transaction involves data that resides on more than one server, such as when a database record is replicated out to two or more servers, MSDTC needs to become involved. SQL Server handles this transparently for you.

However, occasionally, all does not go as well as it should. A server drops off the network or there's a power outage at the distributor server. Something messy that computers aren't very good at dealing with. MSDTC usually handles these scenarios very well, ensuring that the rules involving the database ACID properties are adhered to, so that everything stays in sync as it should, and everyone's happy about the data

in their tables. However, when MSDTC can't recover from one of these scenarios, the SPID which is handling the distributed transaction on one (or more) servers can't do any more work. The result is an orphaned SPID.

In order to mark this as an orphaned, distributed transaction SPID, SQL Server changes the SPID from a positive number to -2. The only problem is the SPID may still be holding on to resources (usually table, page or row locks), and blocking other SPIDs which want to access that database object. Because the KILL command can't handle SPIDs with a value of less than 1, you can't use it to kill a negative SPID. Hence the need to look up the UOW (Unit of Work) ID of the offending SPID before the process can be terminated.

I sincerely hope that this has been of some value to you and it's actually a good and simple way to resolve negative SPID errors. This is a good article for everyone who has faced this issue in the past and a good learning experience for people who may face this issue in future. Please remember that my company does not allow me to participate in forums and websites or blogs on company machine and thus there can be a delay in response.
