


Log Reader Agent How to skip a transaction

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:31 AM PST

Contents

- [Issue](#)
- [Investigation / Analysis](#)
- [Mitigation](#)
- [More Information](#)
 - [Prerequisites used for the following sample steps](#)
 - [Set the Log Reader read batch size to 1](#)
 - [Stop the Log Reader Agent](#)
 - [Add a few changes - test environment only](#)
 - [Identify the Log Sequence Numbers](#)
 - [Move the Distributor start LSN forward](#)
 - [Final steps](#)

Issue

Under specific circumstances, it is possible that the Log Reader Agent gets stuck on reading the next transaction from the transaction log of the published database. The Log Reader can't proceed further, and the transaction log of the published database is growing with the pending replicated changes. The log will continue to grow until either the Log Reader issue is fixed, or the pending transactions are marked as already replicated by calling [sp_repldone](#) .

Possible error messages caused by reading incorrect information from the transaction log are:

Error 18805

The Log Reader Agent failed to construct a replicated command from log sequence number (LSN) {00015b5b:00003230:0139}. Back up the publication database and contact Customer Support Services.

Error 515

Cannot insert the value NULL into column 'article_id', table 'distribution.dbo.MSrepl_commands'; column does not allow nulls. INSERT fails.

This article provides you with the recommended mitigation, but has an alternate solution in the "More Information" section - which is at your own risk though.

Investigation / Analysis

There is nothing you can do to fix this issue without data loss for the Subscribers. There is no direct way to force the Log Reader Agent over the "stuck" transaction, and you can't "repair" the stuck transaction either.

Mitigation

The recommended and preferred solution is to reset the replicated transactions on the transaction log (mark them as replicated), and to reinitialize all subscriptions after creating a new snapshot.

You can execute the following command to mark all transactions as replicated:

```
exec sp_repldone @xactid=NULL, @xact_segno=NULL, @numtrans=0, @time=0, @reset=1
```

This might not be the preferred solution for the customer though, especially if the replication databases are large and reinitializing all subscriptions would take a very long time. If the customer is willing to accept data loss at the Subscribers, they can try the sample steps from the "More information" section below.

More Information

The following steps demonstrate a workaround that skips over the offending failing transaction. The data changes from this transaction are lost. In this example, it is just a simple Insert, but in the customer environment, it might involve a lot of changes for several tables. Skipping this transaction leads to data inconsistencies at the Subscribers, and possibly to further errors with the Distribution Agents. See [Distribution Agent errors 2601 2627 20598](#) for steps to mitigate such later failures.

Disclaimer These steps are provided at your own risk. They might or might not work in a production environment. They will certainly cause data loss at the Subscribers, and they might cause further issues later, specifically Distribution Agent errors 20598 "row not found", 2601 "cannot insert duplicate key", and 2627: "violation of PRIMARY KEY constraint".

Prerequisites used for the following sample steps

(only necessary for your test environment)

Prepare the replication environment as shown in the [Replication Sample Script](#):

- [Configure Distribution](#)
- [Create databases and tables](#)
- [Publish the database](#)
- [Add more sample data](#)

Set the Log Reader read batch size to 1

This step is important to replicate all individual transactions up to the failure. By default, the Log Reader retrieves batches of 500 replicated transactions. If the failure occurs somewhere within those 500 transactions, you want to replicate everything up to the failing transaction to minimize the data loss at the Subscribers.

To do this, change the Log Reader's read batch size to 1, like this:

- Identify the Log Reader Agent job in the SSMS Job Activity Monitor.
- Open the job and go to "Steps".
- Edit the "Run agent" step and add the `-ReadBatchSize 1` parameter to the end of the parameter command line.
- Save the change.

- Restart the agent job and let it run until it fails again.

The ReadBatchSize parameter configures the maximum number of transactions that the Log Reader Agent reads from the transaction log per processing cycle. The default is 500 and by setting it to 1, it single-steps through the transactions up to the one that is causing the failure. The failing transaction will then be the oldest non-distributed transaction and can be clearly identified and skipped with the steps below.

Stop the Log Reader Agent

Stop the Log Reader Agent and disable its schedules, if it has any. This is important to avoid any interference of the agent with the following steps. For your test environment, it will simulate the agent failure so that it doesn't proceed beyond the current log position.

Add a few changes - test environment only

(only necessary for your test environment)

Add a few rows to simulate pending transactions that are waiting to be picked up by the Log Reader:

```
SELECT * from Repl_PUB..MainTable;
-- ID = 1, 2

INSERT INTO [Repl_PUB].[dbo].[MainTable] (ID, c1) VALUES (11, 'To be skipped');
INSERT INTO [Repl_PUB].[dbo].[MainTable] (ID, c1) VALUES (21, 'Row 1 after the skipped row');
INSERT INTO [Repl_PUB].[dbo].[MainTable] (ID, c1) VALUES (22, 'Row 2 after the skipped row');
INSERT INTO [Repl_PUB].[dbo].[MainTable] (ID, c1) VALUES (23, 'Row 3 after the skipped row');
```

The expected outcome is that ID=11 will be skipped later and that ID=21, 22, 23 will be replicated.

Identify the Log Sequence Numbers

Retrieve the current LSNs on Publisher and Distributor.

Note that these will be different in your environment and you must adapt each of the following steps with the results from your environment.

```

use publisherdatabase;

-- Get last distributed transaction
DBCC OPENTRAN
/*
Replicated Transaction Information:
    Oldest distributed LSN      : (51:4080:3)
    Oldest non-distributed LSN : (51:4088:1)
*/

/* Convert LSNs to hexadecimal:
Oldest distributed LSN      Oldest non-distributed LSN
51:4080:3                  51:4088:1
33:FF0:3                   33:FF8:1
*/

-- Acquire logreader context and show the pending replicated transactions
exec sp_repltrans;
/*
xdesid                xact_seqno
0x0000003300000FF80037 0x0000003300000FF8003A  <-- Log reader is here
0x000000330000010000001 0x000000330000010000003
0x000000330000010080001 0x000000330000010080003
0x000000330000010100001 0x000000330000010100003  <-- use this as upper limit for fn_dblog query below
*/
-- release logreader context
exec sp_replflush;

-- Show all transactions starting with the next replicated transaction
-- note the LOP_INSERT_ROWS and how they have Description=REPLICATE
-- note how the xdesid/xact_seqno pairs from sp_repltrans correspond to the LOP_BEGIN_XACT/LOP_COMMIT_XACT pair

-- 0x0000003300000FF80037      0x000000330000010100003
-- 0x00000033:00000FF8:0037    0x00000033:00001010:0003
select [Current LSN], Operation, [Transaction ID], AllocUnitName, Description
from fn_dblog('0x00000033:00000FF8:0037', '0x00000033:00001000:0003');
/*
Current LSN            Operation            Transaction ID  AllocUnitName            Descrip
00000033:00000FF8:0037 LOP_BEGIN_XACT  0000:00001d11  NULL                     2022/12
00000033:00000FF8:0038 LOP_SET_BITS    0000:00000000  dbo.MainTable.PK__MainTabl__3214EC2784E10411 Version
00000033:00000FF8:0039 LOP_INSERT_ROWS 0000:00001d11  dbo.MainTable.PK__MainTabl__3214EC2784E10411 REPLICATION
00000033:00000FF8:003A LOP_COMMIT_XACT 0000:00001d11  NULL                     2022/12
00000033:00001000:0001 LOP_BEGIN_XACT  0000:00001d12  NULL                     2022/12
00000033:00001000:0002 LOP_INSERT_ROWS 0000:00001d12  dbo.MainTable.PK__MainTabl__3214EC2784E10411 REPLICATION
00000033:00001000:0003 LOP_COMMIT_XACT 0000:00001d12  NULL                     2022/12
00000033:00001008:0001 LOP_BEGIN_XACT  0000:00001d13  NULL                     2022/12
00000033:00001008:0002 LOP_INSERT_ROWS 0000:00001d13  dbo.MainTable.PK__MainTabl__3214EC2784E10411 REPLICATION
00000033:00001008:0003 LOP_COMMIT_XACT 0000:00001d13  NULL                     2022/12
00000033:00001010:0001 LOP_BEGIN_XACT  0000:00001d14  NULL                     2022/12
00000033:00001010:0002 LOP_INSERT_ROWS 0000:00001d14  dbo.MainTable.PK__MainTabl__3214EC2784E10411 REPLICATION
00000033:00001010:0003 LOP_COMMIT_XACT 0000:00001d14  NULL                     2022/12

-- The transaction to skip is the first in line:
Current LSN            Operation            Transaction ID
00000033:00000FF8:0037 LOP_BEGIN_XACT  0000:00001d11
00000033:00000FF8:003A LOP_COMMIT_XACT 0000:00001d11

xact_id                xact_seqno
00000033:00000FF8:0037 00000033:00000FF8:003A
0x0000003300000FF80037 0x0000003300000FF8003A
*/

```

Move the Distributor start LSN forward

```

-- Retrieve the publisher_database_id for the Publisher database
-- let the customer point you to the correct database and get its ID
SELECT s.srvname, db.id AS 'publisher_database_id', db.publisher_db, p.publication, p.publication_id
FROM distribution..MSreplservers s
INNER JOIN distribution..MSpublisher_databases db ON s.srv_id = db.publisher_id
INNER JOIN distribution..MSpublications p ON p.publisher_id = db.publisher_id
-- publisher_database_id = 4

-- Search for the initial oldest distributed LSN: 33:FF0:3
--      33      FF0      3
-- 0x0000003300000FF0003
select * from distribution.dbo.MSrepl_transactions
where publisher_database_id = 4 and xact_seqno >= 0x0000003300000FF0003;
/*
publisher_database_id  xact_id                xact_seqno                entry_time
4                      0x0000003300000FF0001    0x0000003300000FF0003    2022-12-15 07:53:49.287
*/

-- Insert the xactid and xact_seqno that you have identified for skipping:
-- xact_id                xact_seqno
-- 0x0000003300000FF80037    0x0000003300000FF8003A

-- use transaction to allow for errors
begin tran

-- we need to set the entry_time to the same as the skipped transaction
-- (if entry_time would be much newer, it might interfere with distribution cleanup later)
declare @dt1 datetime
select @dt1 = entry_time from distribution.dbo.msrepl_transactions
where publisher_database_id = 4 and xact_seqno = 0x0000003300000FF0003

INSERT INTO distribution.dbo.MSrepl_transactions (publisher_database_id, xact_id, xact_seqno, entry_time)
VALUES (4, 0x0000003300000FF80037, 0x0000003300000FF8003A, @dt1);

-- check outcome before committing:
select * from distribution.dbo.MSrepl_transactions
where publisher_database_id = 4 and xact_seqno >= 0x0000003300000FF0003;
/*
publisher_database_id  xact_id                xact_seqno                entry_time
4                      0x0000003300000FF0001    0x0000003300000FF0003    2022-12-15 07:53:49.287
4                      0x0000003300000FF80037    0x0000003300000FF8003A    2022-12-15 07:53:49.287    <-- inserted
*/
-- if it looks OK then commit, otherwise: rollback
commit

```

Final steps

```
-- Remove the ReadBatchSize parameter from the "Run Agent" step of the Log Reader which you had added above
-- Re-enable the job schedule if you had disabled it.

-- Start Log Reader Agent job
-- Start Distribution Agent job if it was stopped

-- Check the results of the synchronization
SELECT * from Repl_PUB..MainTable;
/*
ID    c1
---  -----
1     original insert
2     second insert
11    To be skipped
21    Row 1 after the skipped row
22    Row 2 after the skipped row
23    Row 3 after the skipped row
*/

SELECT * from Repl_SUB..MainTable;
/*
ID    c1
---  -----
1     original insert
2     second insert
21    Row 1 after the skipped row
22    Row 2 after the skipped row
23    Row 3 after the skipped row
*/

/** End of script **/
```

How good have you found this content?



-