

Initialize Subscription from a Backup (MI to MI PITR)

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

Contents

- [Issue](#)
- [Investigation / Analysis](#)
- [Mitigation](#)
 - [Set publication options](#)
 - [Identify the start LSN for the sync](#)
 - [Verify LSN and identify the datetime for the PITR](#)
 - [Restore the Publisher database into a new database](#)
 - [Create subscription using the restored database as Subscri...](#)
 - [Quality Check](#)
- [Public Doc References](#)

Initialize Transaction Replication subscription from a Point-in-time-restored backup (Managed Instance to Managed Instance)

Use INITIALIZE FROM LSN with a PITR backup

This article provides you with steps for initializing a Managed Instance transactional replication subscription with a point-in-time-restored (PITR) backup of the published database. The Publisher and Distributor is hosted on Managed Instance, and the Subscriber is either hosted on the same or a different Managed Instance. Instead of creating and applying a snapshot to the Subscriber, the steps will point-in-time-restore a backup of the Publisher database and create the subscription based on a Log Sequence Number (LSN).

It is also available publicly at [Initialize a Managed Instance replication Subscriber with a PITR backup](#) .

Issue


Although a subscription to a transactional publication is typically initialized with a snapshot, it might take a very long time for doing so. To reduce the time until the Subscriber becomes available, a subscription can be initialized from a backup using replication stored procedures.

This article describes the steps for initializing a Managed Instance subscription from a restored Managed Instance published database backup.

NOTE 1: The steps describe a general approach that you can try out on your own Managed Instance. The steps have been tested successfully on a non-production environment, but there is no guarantee that they will work for all situations and all environments. You still need to work out the exact LSN details and command parameter details for yourself. Please share your experience in the comments below if you find any flaws or have ideas for improvement.

NOTE 2: This article only applies to replicating from Managed Instance to Managed Instance. It does NOT apply to replicating from/to on-premise SQL Server. For the on-premise to MI scenario, please see article [Initialize Subscription from a Backup \(avoid using Snapshot\)](#).

Investigation / Analysis

The `@sync_type` parameter of the [sp_addsubscription \(Transact-SQL\)](#)  stored procedure indicates how you initialize the Subscriber database. There are four options available:

- `automatic` is the default option of applying a snapshot to the Subscriber.
- `replication support only` assumes that the Subscriber already has the schema and data. This method only adds metadata and replication stored procedures to the Subscriber database, and assumes that the data is 1:1 the same and unchanged from the Publisher database. It usually requires that the Publisher is quiesced while the data and subscription is created at the Subscriber.
- `initialize with backup` would be the expected option for this topic. It doesn't work for MI databases though, because it requires you to provide a path to the backup files. But you do not have access to the PITR media, and creating your own `copy_only` backup often fails on the TDE restrictions.
- `initialize from lsn` is an option for adding nodes to a Peer-to-Peer Transactional Replication topology. Although Peer-to-Peer is not supported on Managed Instance, the `initialize from lsn` option is still available on MI. You have to provide a valid start Log Sequence Number (LSN) in the `@subscription_lsn` parameter, which needs to be included in the restored database. *This is the approach chosen for the steps below.*

The trade-off between these options is the time it takes to create the snapshot and applying it to the Subscriber, vs. the time it takes to point-in-time restore the database. The snapshot might still be the better option, especially if not all tables in the database are published and the unpublished data is much larger than the published data.

Mitigation

The following steps are complicated to follow, and they might or might not work for you. If you get them to work, they will provide you with a great time-saving option when initializing large Subscriber databases.

The difficulty is to identify a valid LSN and the correct point in time for the restore. The best way is to quiesce the Publisher database for a short time, e.g. 15 or 30 minutes, to help you identifying a clear starting point for restoring the database and creating the subscription.

The sample steps below contain actual values from a test environment, e.g. the publication name and the LSNs, to demonstrate the process. You will see different values on your environment and have to adapt the steps accordingly. If possible, try it out on a simple test environment before applying it to a complex production environment.

Set publication options

Verify that the publication has the `immediate_sync` and `allow_initialize_from_backup` options enabled:

```
-- on the PUBLISHER database
select allow_initialize_from_backup, immediate_sync, immediate_sync_ready, min_autonosync_lsn, name from syspub
/*
allow_initialize_from_backup immediate_sync immediate_sync_ready min_autonosync_lsn      name
1                          1              1                  0x000000A0000009F8003E  Publication_Tran
*/

-- should return 1 for "allow_initialize_from_backup" and "immediate_sync"
```

If either of "allow_initialize_from_backup" or "immediate_sync" is disabled = 0, enable them:

```
-- on the PUBLISHER database
EXEC sp_changepublication
    @publication = 'Publication_Tran',
    @property = 'immediate_sync',
    @value = 'true'
GO
EXEC sp_changepublication
    @publication = 'Publication_Tran',
    @property = 'allow_initialize_from_backup',
    @value = 'true'
GO
```

If you had to update these options, you might have to run the Snapshot Agent once to set and refresh the internal LSNs on the metadata accordingly.

Identify the start LSN for the sync

Retrieve the start LSN from the Publisher and Distribution databases. The "virtual" subscriber_db is referring to the "immediate_sync" feature and maintains a common LSN that is available in the recent snapshot and in the distribution database:

```
-- on the PUBLISHER database
select allow_initialize_from_backup, immediate_sync, immediate_sync_ready, min_autonosync_lsn, name from syspub
/*
allow_initialize_from_backup immediate_sync immediate_sync_ready min_autonosync_lsn      name
1                          1              1                  0x000000A0000009F8003E  Publication_Tran
*/

--on the DISTRIBUTION database
select allow_initialize_from_backup, immediate_sync, min_autonosync_lsn from distribution.dbo.mspublications
/*
allow_initialize_from_backup immediate_sync min_autonosync_lsn      publisher_id publisher_db publication
1                          1              0x000000A0000009F8003E  1          Repl_PUB      Publication_T
*/

--on the DISTRIBUTION database
select publisher_database_id, publisher_db, subscriber_id, subscriber_db, article_id, subscription_seqno, publ
from distribution.dbo.mssubscriptions
where subscriber_db = 'virtual'
/*
publisher_database_id publisher_db subscriber_id subscriber_db article_id subscription_seqno
1                    Repl_PUB      -1          virtual      1          0x000000A0000009F8004500000001
1                    Repl_PUB      -1          virtual      2          0x000000A0000009F8004500000001
1                    Repl_PUB      -2          virtual      1          0x000000A0000009F8004500000001
1                    Repl_PUB      -2          virtual      2          0x000000A0000009F8004500000001
*/
```

The LSN that should always work is the "minimum auto-nosync LSN" (`min_autonosync_lsn`) from the `syspublications/mspublications` system tables. But any of the other LSNs (`subscription_seqno/publisher_seqno/ss_cplt_seqno`) should also work accordingly. Or you might use any other LSN, as long as you can correlate a point in time for the restore to it. The key is to identify an LSN that is available in the restored backup and the distribution database.

For the next steps in this example, the `min_autonosync_lsn` value of "0x000000A0000009F8003E" will be used.

Verify LSN and identify the datetime for the PITR

For verification, search directly for the chosen LSN in the existing replicated transactions. You may find it in the `MSrepl_transactions` system table, either in column `xact_id` or `xact_seqno`:

```
--on the DISTRIBUTION database
--filter on publisher_database_id if the output is too large
select * from distribution..MSrepl_transactions
where xact_id >= 0x000000A0000009F8003E or xact_seqno >= 0x000000A0000009F8003E

/*
publisher_database_id  xact_id                xact_seqno                entry_time
1                      0x0000000000000000      0x000000A0000009F8003E00000001  2022-06-03 13:16:53.090
1                      0x0000000000000000      0x000000A0000009F8003E00000002  2022-06-03 13:16:53.090
1                      0x0000000000000000      0x000000A0000009F8003E00000003  2022-06-03 13:16:53.090
1                      0x0000000000000000      0x000000A0000009F8003E00000004  2022-06-03 13:16:53.090
1                      0x0000000000000000      0x000000A0000009F8003E00000005  2022-06-03 13:16:53.090
1                      0x000000A0000009F80001  0x000000A0000009F80045        2022-06-03 13:16:53.090
1                      0x00                0x000000A0000009F8004500000001  2022-06-03 13:18:30.543
(...)
*/
```

The `entry_time` column will show you the possible point in times for the database restore. In this example, the best time would be "2022-06-03 13:16:53" or "2022-06-03 13:16:54": the `min_autonosync_lsn` of 0x000000A0000009F8003E is falling into the range of 0x000000A0000009F80001...0x000000A0000009F80045, and there is a distinct time gap between this and the next transaction, 2022-06-03 13:16:53.090 and 2022-06-03 13:18:30.543.

To get a better understanding of the workload at that time, you should also look into the vicinity of that LSN and time. If the replication environment was very busy at that time, it might not be easy to correlate the restore time with a specific LSN. Therefore quiescing the replication for some time (15 or 30 minutes) will give you an easy way for identifying a start LSN.

--on the DISTRIBUTION database
 --filter on publisher_database_id if the output is too large

```
select * from distribution..MSrepl_transactions
order by xact_seqno
/*
publisher_database_id  xact_id                xact_seqno                entry_time
(...)
1                      0x00                      0x0000008500000970000500000005  2022-05-06 13:19:02.523
1                      0x00                      0x0000008500000970000500000006  2022-05-06 13:26:28.650
1                      0x00000000000000          0x000000A0000009F8003E00000001  2022-06-03 13:16:53.090
1                      0x00000000000000          0x000000A0000009F8003E00000002  2022-06-03 13:16:53.090
1                      0x00000000000000          0x000000A0000009F8003E00000003  2022-06-03 13:16:53.090
1                      0x000000A0000009F80001    0x000000A0000009F80045          2022-06-03 13:16:53.090
1                      0x00                      0x000000A0000009F8004500000001  2022-06-03 13:18:30.543
1                      0x00000000000000          0x000000A000000AE0000100000001  2022-06-03 13:43:11.103
1                      0x00000000000000          0x000000A000000AE0000100000002  2022-06-03 13:43:11.103
1                      0x000000A000000BE00002    0x000000A000000BF00006          2022-06-03 13:36:25.127
1                      0x000000A000000C380005    0x000000A000000C380009          2022-06-03 13:43:11.103
1                      0x000000A3000006A80044    0x000000A3000006A8004A          2022-06-07 14:25:44.873
1                      0x000000A3000006B80003    0x000000A3000006B80006          2022-06-07 14:28:15.147
1                      0x000000A3000006C00001    0x000000A3000006C00003          2022-06-07 14:28:15.147
(...)
*/

select * from distribution..MSrepl_transactions
order by entry_time
/*
publisher_database_id  xact_id                xact_seqno                entry_time
(...)
1                      0x00                      0x0000008500000970000500000004  2022-05-04 12:13:15.973
1                      0x00                      0x0000008500000970000500000005  2022-05-06 13:19:02.523
1                      0x00                      0x0000008500000970000500000006  2022-05-06 13:26:28.650
1                      0x00000000000000          0x000000A0000009F8003E00000001  2022-06-03 13:16:53.090
1                      0x00000000000000          0x000000A0000009F8003E00000002  2022-06-03 13:16:53.090
1                      0x00000000000000          0x000000A0000009F8003E00000003  2022-06-03 13:16:53.090
1                      0x000000A0000009F80001    0x000000A0000009F80045          2022-06-03 13:16:53.090
1                      0x00                      0x000000A0000009F8004500000001  2022-06-03 13:18:30.543
1                      0x000000A000000BE00002    0x000000A000000BF00006          2022-06-03 13:36:25.127
1                      0x000000A000000C380005    0x000000A000000C380009          2022-06-03 13:43:11.103
1                      0x00000000000000          0x000000A000000AE0000100000001  2022-06-03 13:43:11.103
1                      0x00000000000000          0x000000A000000AE0000100000002  2022-06-03 13:43:11.103
1                      0x000000A3000006A80044    0x000000A3000006A8004A          2022-06-07 14:25:44.873
1                      0x000000A3000006B80003    0x000000A3000006B80006          2022-06-07 14:28:15.147
(...)
*/
```

In this case here, it confirms the `entry_time` gap between 13:16:53.090 and 13:18:30. Let's set a PITR time that is slightly inside the gap, "2022-06-03T13:16:54".

These times were collected from an instance that is configured with UTC time. If you had chosen a custom time zone for your MI, you might have to adapt this to UTC for the restore and play with this a bit.

Restore the Publisher database into a new database

After having chosen the LSN and the exact time for the point-in-time restore, the next step is the actual restore. You will restore the Publisher database into a new, separate database, which will act as Subscriber database. You may restore the database either to the same instance, or to a different instance that the Publisher/Distributor can connect to, usually an MI on the same subscription.

The portal might not honour the exact second that you enter for the restore operation, but PowerShell does. Therefore the restore will use the `Restore-AzSqlInstanceDatabase` PowerShell command as described in: [Restore](#)

[an existing database](#) 

```
## restore: -- 0x000000A0000009F8003E 2022-06-03 13:16:53.090
## contains test parameter values to demonstrate the exact syntax

## set both instance names to the same name for restoring to the same MI
```

Connect-AzAccount

```
$subscriptionId = "e11074c4-0d8c-491e-bd96-cd40ef914e22"
$resourceGroupName = "publisher-rg"
$managedInstanceName = "replsrv1"
$databaseName = "Repl_PUB"
$targetResourceGroupName = "subscriber-rg"
$targetInstanceName = "replsrv2"
$targetDatabase = "Repl_PUB-restore"

$pointInTime = "2022-06-03T13:16:54Z"
```

Set-AzContext -SubscriptionId \$subscriptionId

```
Restore-AzSqlInstanceDatabase -FromPointInTimeBackup `
    -ResourceGroupName $resourceGroupName `
    -InstanceName $managedInstanceName `
    -Name $databaseName `
    -PointInTime $pointInTime `
    -TargetInstanceDatabaseName $targetDatabase `
    -TargetResourceGroupName $targetResourceGroupName `
    -TargetInstanceName $targetInstanceName
```

Create subscription using the restored database as Subscriber

Before running this step, you need to make sure that the Distribution Agent can connect to the Subscriber and into the Subscriber database. After restoring the database, adjust the SQL logins and database users at the Subscriber to make sure that the following call to `sp_addpushsubscription_agent` can use a valid user/password for the job login. The user should preferably be a login at the target server, with db_owner permissions in the target Subscriber database.

Then create the new subscription - specify the name of the restored database as target and set the `subscriptionlsn` parameter to the LSN that you have identified in the steps above:

```
-- on the PUBLISHER database
-- contains test parameter values to demonstrate the exact syntax

-- Create subscription at the Publisher using 'initialize from lsn' and specifying the LSN from above:
exec sp_addsubscription @publication = 'Publication_Tran',
@subscriber = 'replsrv2.8f08e6d34d3b.database.windows.net', @destination_db = 'Repl_PUB-restore',
@sync_type = 'initialize from lsn',
@subscription_lsn = 0x000000A0000009F8003E,
@subscription_type = N'Push',
@article = N'all',
@update_mode = N'read only',
@subscriber_type = 0

-- Configure the Distribution Agent
exec sp_addpushsubscription_agent @publication = N'Publication_Tran',
    @subscriber = 'replsrv2.8f08e6d34d3b.database.windows.net', @subscriber_db = N'Repl_PUB-restore',
    @job_login = 'TRANREPLADMIN', @job_password = '$trongPa11word',
    @subscriber_security_mode = 0, @subscriber_login = 'TRANREPLADMIN', @subscriber_password = '$trongPa11
GO
```

Quality Check

Go into Replication Monitor and confirm that the Distribution Agent is picking up the current changes and that there are no errors.

If you picked a wrong LSN or a wrong time for the PITR, then the Distribution Agent might fail with an error 20598 "row not found" or a Primary Key violation (errors 2601/2627).

For troubleshooting, you could then execute `sp_browsereplcmds` in the Distribution database to get a clear-text overview on the pending commands. Compare the details from the error message with the pending commands. Then adjust either the start LSN or the entry_time and try again.

Public Doc References

- [Restore a database in Azure SQL Managed Instance to a previous point in time](#) 

How good have you found this content?



-