


Initialize Subscription from a Backup (on-premise to MI)

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

Contents

- [Initialize with Backup sample script](#)
 - [Introduction](#)
 - [Prerequisites:](#)
 - [Section 1 - Preparing the on-premise SQL Server for Trans...](#)
 - [Section 2 - Create the publication on the on-premise SQL ...](#)
 - [Section 3 - Backup the Publisher database](#)
 - [Section 4 - Restore the backup to the Subscriber](#)
 - [Section 5 - Create the subscription and initialize with backup](#)
 - [Bonus Section - Create a subscription on the on-premise S...](#)
 - [Public Doc References](#)

Initialize a Subscriber with a backup (Publisher/Distributor on-premise, Subscriber Managed Instance)

The content of this article is "customer-ready" and can be shared with external customers as needed. It has also been published in a [public blog article](#)  with the same content.

Initialize with Backup sample script

This article provides you with a sample script for creating a transactional replication with the Publisher and Distributor hosted on-premise and the Subscriber being a Managed Instance database. Instead of creating and applying a snapshot to the Subscriber, the script will show you the steps to initialize the Subscriber with a backup of the Publisher database.

The script keeps details as simple and straightforward as possible to demonstrate the replication feature itself, and to allow for a quick setup to test specific functionality. We are using this type of script in our support work to reproduce customer issues and test the behaviour of specific configuration options.

Introduction




A subscription to a transactional publication is by default initialized with a snapshot. The snapshot is generated by the Snapshot Agent and applied by the Distribution Agent, which is reasonably fast in most on-premise environments. But snapshots can run into significant delays if the published dataset is large or if the Distributor and Subscriber are connected over slow networks.

Slow Subscriber connectivity can occur if the on-premise SQL Server needs to publish its data to a Managed Instance Subscriber database, where the internet upload link only has limited throughput. In the worst case, it

might take hours to create the large snapshot, and then several days for applying the snapshot over the internet into the cloud database; it might even exceed the default retention period of 72 hours = 3 days.

This issue can be avoided by initializing the Subscriber database with a backup of the Publisher database. Initializing with a backup is the fastest way to deliver data to the Subscriber and is convenient, because any recent backup can be used if it was taken after the publication was enabled for initialization with a backup.

Prerequisites:

- This article assumes that you are already familiar with Managed Instance, especially its connectivity requirements.
- It assumes that you know the basics of Transactional Replication and how to configure Transactional Replication.
- You need to have an Azure Blob Storage container and its credentials available for uploading the backup file.
- You need a SQL Server on-premise instance, either hosted on a physical machine or in a virtual machine, including an Azure VM.
- The on-premise SQL Server must be able to connect to the Managed Instance; it requires [point-to-site](#)  (see [step-by-step guide](#) ) or [ExpressRoute](#)  connectivity.
- The server name (@@SERVERNAME) of the on-premise SQL Server returns its NETBIOS name.
- The server name (@@SERVERNAME) of the Managed Instance returns its fully-qualified domain name (FQDN).

Section 1 - Preparing the on-premise SQL Server for Transactional Replication

Run this section on the on-premise SQL Server - it includes the following steps:

- enables Distribution on SQL Server

```

/*****
-- Transaction Replication INITIALIZE WITH BACKUP sample script
-- needs a SQL Server instance and a Managed Instance
-- the SQL Server will act as Publisher, Distributor and Subscriber
-- The MI will act as Subscriber to the SQL Server instance
-- Both subscribers will be initialized from a backup of the Publisher database
*****/

/*****
/**** SECTION 1 ****/
/**** run this section at the on-premise SQL Server ****/
*****/

-- Add the distributor and the distribution database.
USE [master]
GO
-- check if distribution has already been configured:
exec sp_get_distributor
GO
-- check server name to verify to be connected to the correct server
select @@servername
GO

-- Fill in the name of the Distributor instance as needed:
-- The password is the "Administrative Link password" that you also see on the Distributor Properties page
EXEC sp_adddistributor @distributor = @@ServerName, @password = N'$trongPa11word';
GO
EXEC sp_adddistributiondb @database = N'Distribution';
GO

-- Configure the remote Publisher at the Distributor
EXEC sp_adddistpublisher @publisher = @@ServerName,
    @distribution_db = N'Distribution',
    @security_mode = 1,
    @working_directory = N'M:\Snapshots\'
GO

-- confirm that distribution has been configured:
exec sp_get_distributor
GO

/**** END OF SECTION 1 *****/

```

Section 2 - Create the publication on the on-premise SQL Server

Run this section of the script at the on-premise SQL Server only after section 1 has completed successfully. It requires that the publisher database has already been created and marked as published.

Note the command `exec sp_addpublication ... @allow_initialize_from_backup = N'true' ... @immediate_sync = N'true'`. These options are required to mark the start Log Sequence Number in the transaction log when the backup is created, and to keep the pending changes on the transaction log until the backup has been successfully restored and the subscription created.

This section of the script includes the following steps:

- creates a database and marks it as published for Transactional Replication
- configures the Log Reader Agent
- creates sample tables with test data
- creates the publication and articles

```

/*****
/**** SECTION 2 - run this script at the on-premise SQL Server ****
/****
-- Create the publisher database
SET NOCOUNT ON
GO
USE [master]
GO
CREATE DATABASE [Repl_PUB]
GO

-- Enable the replication database for publication
USE [master]
GO
exec sp_replicationdboption @dbname = N'Repl_PUB', @optname = N'publish', @value = N'true'
GO
-- configure the log reader agent
exec [Repl_PUB].sys.sp_addlogreader_agent @job_login = null, @job_password = null, @publisher_security_mode =
GO

-- Create the sample table and add a few rows
USE [Repl_PUB]
GO
CREATE TABLE [dbo].[MainTable] (
    [ID] [int] PRIMARY KEY CLUSTERED NOT NULL ,
    [c1] [varchar](100) NULL
)
GO
-- DROP TABLE [dbo].[SubTable]
CREATE TABLE [dbo].[SubTable] (
    [ID] [int] PRIMARY KEY CLUSTERED NOT NULL,
    [MainTableID] [int] NOT NULL,
    [c1] [varchar](100) NULL
    , CONSTRAINT FK_Main_Sub FOREIGN KEY (MainTableID) REFERENCES dbo.MainTable (ID) --NOT FOR REPLICATION
)
GO

INSERT INTO [dbo].[MainTable] (ID, c1) VALUES (1, 'original insert')
INSERT INTO [dbo].[SubTable] (ID, MainTableID, c1) VALUES (1, 1, 'original insert')
GO
SELECT * FROM MainTable
SELECT * FROM SubTable
GO

-- Add Publication
use [Repl_PUB]
exec sp_addpublication @publication = N'Publication_Tran', @description = N'Simple Transactional publication',
    @sync_method = N'native', @retention = 0,
    @allow_push = N'true', @allow_pull = N'true', @allow_anonymous = N'true', @enabled_for_internet = N'false'
    @snapshot_in_defaultfolder = N'true', @compress_snapshot = N'false', @ftp_port = 21, @ftp_login = N'anony
    @allow_subscription_copy = N'false', @add_to_active_directory = N'false',
    @repl_freq = N'continuous', @status = N'active', @independent_agent = N'true',
    @immediate_sync = N'true',
    @allow_sync_tran = N'false', @autogen_sync_procs = N'false', @allow_queued_tran = N'false', @allow_dts =
    @replicate_ddl = 1, @allow_initialize_from_backup = N'true', @enabled_for_p2p = N'false', @enabled_for_he
GO

-- configure the Snapshot agent
-- this is not needed because we initialize with backup later

-- Grant permissions to publication
exec sp_grant_publication_access @publication = N'Publication_Tran', @login = N'sa'
GO

-- Add the articles

```

```

use [Repl_PUB]
exec sp_addarticle @publication = N'Publication_Tran',
    @article = N'MainTable', @source_owner = N'dbo', @source_object = N'MainTable', @destination_table = N'Ma
    @type = N'logbased', @description = N'', @creation_script = N'',
    @pre_creation_cmd = N'drop',
    @schema_option = 0x000000000803529F, @identityrangemanagementoption = N'none',
    @status = 24, @vertical_partition = N'false',
    @ins_cmd = N'CALL [dbo].[sp_MSins_dboMainTable]', @del_cmd = N'CALL [dbo].[sp_MSdel_dboMainTable]', @upd_
GO
exec sp_addarticle @publication = N'Publication_Tran',
    @article = N'SubTable', @source_owner = N'dbo', @source_object = N'SubTable', @destination_table = N'SubT
    @type = N'logbased', @description = N'', @creation_script = N'',
    @pre_creation_cmd = N'drop',
    @schema_option = 0x000000000803529F, @identityrangemanagementoption = N'none',
    @status = 24, @vertical_partition = N'false',
    @ins_cmd = N'CALL [dbo].[sp_MSins_dboSubTable]', @del_cmd = N'CALL [dbo].[sp_MSdel_dboSubTable]', @upd_cm
GO

/** END OF SECTION 2 *****/

```

Section 3 - Backup the Publisher database

OPTION 1 Works with all Publisher SQL Server versions

Although it is possible to backup the database directly to an Azure Blob storage URL, older SQL Server versions cannot use it for initializing the Subscriber from a backup because the [sp_addsubscription \(Transact-SQL\)](#) ☐ command doesn't allow the `backupdevicetype = 'URL'`. Older SQL versions must use a local storage path for writing the backup file, and you then have to upload the file to a Blob storage container yourself to make it available to the MI restore command. Note the `COMPRESSION` flag to minimize the file size for quicker upload to Azure, and the `COPY_ONLY` flag to avoid breaking the backup chain. This option is still valid on newer SQL versions, e.g. if the SQL Server machine doesn't have direct access to Azure storage.

OPTION 2 Applies only to Publisher SQL Server versions 2017 CU21 / 2019 CU7 / 2022 and newer

The Cumulative Update described in [KB4569425 - FIX: Transactional replication publications can support URL type device](#) ☐ allows using the `backupdevicetype = 'URL'` for the [sp_addsubscription \(Transact-SQL\)](#) ☐ command. This saves the manual step for uploading the backup file to an Azure storage container.

Run this section of the script at the on-premise SQL Server, and run it only after Section 2 has completed successfully. You need to have the publication in place before creating the database backup.

```

/*****
/*** SECTION 3 - run this script at the on-premise SQL Server ***
/*** run this only after Section 2 has completed successfully ***
/*****/

-- Create a backup of the publication database

/**** OPTION 1 ****/
BACKUP DATABASE [Rep1_PUB] TO DISK = 'M:\Backups\Rep1_PUB.bak' WITH COPY_ONLY, INIT, COMPRESSION
GO

-- Copy the backup file to an Azure Blob Storage container
-- use the portal upload option or any other method
-- the backup needs to be available in a Blob container for the restore

/**** OPTION 2 ****/
-- Applies only to Publisher SQL Server versions 2017 CU21 / 2019 CU7 / 2022 and newer

-- Create a credential for the Blob storage container if you haven't already
USE master
CREATE CREDENTIAL [https://youraccount.blob.core.windows.net/backups]
WITH
    IDENTITY='SHARED ACCESS SIGNATURE',
    SECRET = 'sv=2021-06-08&ss=bfqt&srt=sco&sp=rwdlacupiytfx&se=2024-11-30T...xxxx...Fw0%3D'
GO

BACKUP DATABASE [Rep1_PUB] TO URL = N'https://youraccount.blob.core.windows.net/backups/Rep1_PUB.bak'

/**** END OF SECTION 3 *****/

```

Section 4 - Restore the backup to the Subscriber

Restore the backup from the Azure storage URL to the Subscriber server. Note that you must have a credential based on an SAS token in place to get the permissions to the Blob container. See [Tutorial: Use Azure Blob Storage with SQL Server 2016](#) for more information.

Run this section of the script at the Managed Instance, and run it only after Section 3 has completed successfully. You need to have the database backup available in the Blob storage container for the restore to work.

This section of the script includes the following steps:

- Create a credential for the Blob storage container
- Restore the backup into the Subscriber database
- Create a SQL login and user on the Subscriber that will be used later by the Push Distribution Agent

```

/*****
/** SECTION 4 - run this script at the Managed Instance *****/
/** run this only after Section 3 has completed successfully ***/
*****/

-- Create a credential for the Blob storage container if you haven't already
USE master
CREATE CREDENTIAL [https://youraccount.blob.core.windows.net/backups]
WITH
    IDENTITY='SHARED ACCESS SIGNATURE',
    SECRET = 'sv=2021-06-08&ss=bfqt&srt=sco&sp=rwdlacupiytfx&se=2024-11-30T...xxxx...Fw0%3D'
GO

-- Restore the backup on the MI Subscriber
-- run this from a SSMS connection to the target MI
-- credential to the URL needs to be in place before running this
RESTORE DATABASE [Repl_SUB_onprem] FROM URL = 'https://youraccount.blob.core.windows.net/backups/Repl_PUB.bak'
GO

/* Variation if you want to use the same backup for an on-premise Subscriber:
-- same approach for the on-premise restore - requires to include the MOVE option:
RESTORE DATABASE [Repl_SUB_onprem] FROM URL = 'https://youraccount.blob.core.windows.net/backups/Repl_PUB.bak'
    WITH MOVE 'Repl_PUB' to 'M:\data\Repl_SUB.mdf',
    MOVE 'Repl_PUB_Log' to 'M:\data\Repl_SUB.ldf', REPLACE
GO
*/

-- Create a SQL Login for the replication agents
USE [master]
GO
CREATE LOGIN TRANREPLADMIN WITH PASSWORD = '$trongPa11word';
GO

USE [Repl_SUB_onprem]
GO
CREATE USER TRANREPLADMIN FROM LOGIN TRANREPLADMIN
GO
EXEC sp_addrolemember N'db_owner', N'TRANREPLADMIN'
GO
USE [master]
GO

/**** END OF SECTION 4 *****/

```

Section 5 - Create the subscription and initialize with backup

After restoring the backup from the Azure storage URL, you can now create the Push subscription at the on-premise Distributor.

This section of the script includes the following steps:

- adds a Push subscription at the on-premise Distributor into the Subscriber server and database
- configures the Distribution Agent with Subscriber login details and an execution schedule
- increases the login timeouts of the replication agents to 150 seconds (the default of 15 seconds is too short)
- starts the Distribution Agent

OPTION 1 Works with all Publisher SQL Server versions

Note that the `backupdevicename` uses the same local storage path to which the backup had been written initially

- for older SQL versions you cannot specify the URL on Azure storage; in general you also cannot specify any other, random path.

OPTION 2 Applies only to Publisher SQL Server versions 2017 CU21 / 2019 CU7 / 2022 and newer

The newer SQL versions allow using `backupdevicetype = 'URL'` which simplifies the overall steps.


```

/*****
/** SECTION 5 - run this script at the on-premise SQL Server */
/** run this only after Section 4 has completed successfully */
*****/

/** OPTION 1 */
-- Adding the Push subscription from on-premise to MI
exec [Repl_PUB].sp_addsubscription @publication = N'Publication_TRAN',
    @subscriber = 'yourmi.8f08e6d34d3b.database.windows.net',
    @destination_db = N'Repl_SUB_onprem',
    @subscription_type = N'Push',
    @sync_type = N'initialize with backup',
    @backupdevicetype = N'disk',
    @backupdevicename = N'M:\Backups\Repl_PUB.bak',
    @article = N'all',
    @update_mode = N'read only',
    @subscriber_type = 0
GO

/** OPTION 2 */
-- Adding the Push subscription from on-premise to MI
exec [Repl_PUB].sp_addsubscription @publication = N'Publication_TRAN',
    @subscriber = 'yourmi.8f08e6d34d3b.database.windows.net',
    @destination_db = N'Repl_SUB_onprem',
    @subscription_type = N'Push',
    @sync_type = N'initialize with backup',
    @backupdevicetype = N'URL',
    @backupdevicename = N'https://youraccount.blob.core.windows.net/backups/Repl_PUB.bak',
    @article = N'all',
    @update_mode = N'read only',
    @subscriber_type = 0
GO

-- Configure the Distribution Agent job
exec [Repl_PUB].sp_addpushsubscription_agent @publication = N'Publication_TRAN',
    @subscriber = 'yourmi.8f08e6d34d3b.database.windows.net',
    @subscriber_db = N'Repl_SUB_onprem',
    @job_login = NULL, @job_password = NULL,
    @subscriber_security_mode = 0,
    @subscriber_login = 'TRANREPLADMIN', @subscriber_password = '$trongPa1lword',
    @frequency_type = 64, @frequency_interval = 0, @frequency_relative_interval = 0, @frequency_recurrence_fa
--
GO
    @frequency_type = 4, @frequency_interval = 1, @frequency_relative_interval = 0, @frequency_recurrence_f

-- you might see initial connectivity issues because the default login timeout is much too low
-- Increase the default login timeouts for all jobs
update msdb..sysjobsteps set command = command + N' -LoginTimeout 150'
where subsystem in ('Distribution','LogReader','Snapshot') and command not like '%-LoginTimeout %'

-- Start Distribution agent
exec Repl_PUB.sp_startpushsubscription_agent
    @publication = N'Publication_Tran',
    @subscriber = 'yourmi.8f08e6d34d3b.database.windows.net',
    @subscriber_db = N'Repl_SUB_onprem'
GO

-- Add new sample data into the publisher
-- you also can run this from a SQL Agent job e.g. every 1 minute
declare @i int, @val as varchar(100)
select @i = max(ID) + 1, @val = CONVERT(nvarchar(30), GETDATE(), 121) from dbo.MainTable
INSERT INTO [dbo].[MainTable]([ID], [c1]) VALUES (@i, @val)

-- Check Replication Monitor here

-- Check the results of the synchronization
-- on publisher:
SELECT TOP 5 * from Repl_PUB.dbo.MainTable order by ID desc;

```

```

SELECT TOP 5 * from Repl_PUB.dbo.SubTable order by ID desc;
-- on subscriber:
SELECT TOP 5 * from Repl_SUB_onprem.dbo.MainTable order by ID desc;
SELECT TOP 5 * from Repl_SUB_onprem.dbo.SubTable order by ID desc;

```

```

/** END OF SECTION 5 *****/

```

```

/*****/
/** END OF SCRIPT *****/
/*****/

```

Bonus Section - Create a subscription on the on-premise SQL Server for comparison

```

/*****/
/** BONUS SECTION - run this script at the on-premise SQL Server **/
/** run this only after Section 3 has completed successfully ***/
/*****/

-- Create a credential for the Blob storage container if you haven't already
USE master
CREATE CREDENTIAL [https://youraccount.blob.core.windows.net/backups]
WITH
    IDENTITY='SHARED ACCESS SIGNATURE',
    SECRET = 'sv=2021-06-08&ss=bfqt&srt=sco&sp=rwdlacupiytfx&se=2024-11-30T...xxxx...Fw0%3D'
GO

-- Restore the backup to the on-premise SQL Server
-- run this from a SSMS connection to the on-premise SQL Server
-- credential to the URL needs to be in place before running this
-- same approach as the MI restore - but requires to include the MOVE option
RESTORE DATABASE [Repl_SUB_onprem] FROM URL = 'https://youraccount.blob.core.windows.net/backups/Repl_PUB.bak'
    WITH MOVE 'Repl_PUB' to 'M:\data\Repl_SUB.mdf',
    MOVE 'Repl_PUB_Log' to 'M:\data\Repl_SUB.ldf', REPLACE
GO

-- Adding the Push subscription from on-premise to on-premise
exec [Repl_PUB]..sp_addsubscription @publication = N'Publication_TRAN',
    @subscriber = @@SERVERNAME,
    @destination_db = N'Repl_SUB_onprem',
    @subscription_type = N'Push',
    @sync_type = N'initialize with backup',
    @backupdevicetype = N'disk', @backupdevicename = N'M:\Backups\Repl_PUB.bak',
    @article = N'all',
    @update_mode = N'read only',
    @subscriber_type = 0
GO

exec [Repl_PUB]..sp_addpushsubscription_agent @publication = N'Publication_TRAN',
    @subscriber = @@SERVERNAME,
    @subscriber_db = N'Repl_SUB_onprem',
    @job_login = NULL, @job_password = NULL,
    @subscriber_security_mode = 1,
    @frequency_type = 64, @frequency_interval = 0, @frequency_relative_interval = 0, @frequency_recurrence_fa
--
    @frequency_type = 4, @frequency_interval = 1, @frequency_relative_interval = 0, @frequency_recurrence_f
GO

/** END OF BONUS SECTION *****/

```

Public Doc References

- [Transactional Replication](#) 
- [sp_addsubscription \(Transact-SQL\)](#) 
- [Best Practices for Replication Administration](#) 

How good have you found this content?



-