*MSync Configuration*

Version 1.0

Table of Contents

[1 Purpose 4](#_Toc484178285)

[2 Document Location 4](#_Toc484178286)

[3 Publisher Database 4](#_Toc484178287)

[4 Subscriber Database 4](#_Toc484178288)

[5 Distribution 5](#_Toc484178289)

[6 Adding Tables 6](#_Toc484178290)

[7 Removing Tables 6](#_Toc484178291)

[8 Maintenance 6](#_Toc484178292)

# Purpose

The purpose of this document is to provide instructions for configuring MSync Replication.

# Package Location

\\MSync

# Publisher Database

Follow these steps to configure the Publisher database:

1. Check if the schema rpl already exists. If not, apply script: [1 publisher code.sql]  
   Be careful NOT to reapply the script, or the metadata tables will be wiped out.
2. For each table:

exec rpl.spPublishTable 'SchemaName','TableName'

Note 1: This command adds the rv column to the table, which is a blocking operation and may take several minutes depending on the number of rows, whether the table is replicated or the database is in high availability mode (Mirror and AlwaysOn).

Note 2: For a complete list of all existing subscriptions and import logs please refer to the troubleshooting guide.

# Subscriber Database

Follow these steps to configure the Subscriber database:

1. Check to see if the schema rpl already exists. If not, apply script: [2 subscriber code.sql]  
   Be careful NOT to reapply the script, or the metadata tables will be wiped out.
2. Check if a subscription already exists (select \* from rpl.Subscription). If not, create Subscription and take note of the generated SubscriptionId.

exec rpl.spCreateSubscription @Name = 'FriendlyName', @PriorityGroup = 1, @Server = 'PublisherServer', @Database = 'PublisherDatabase', @Frequency = 1

Note 1: By default, every subscription is created with flags Active=0 and Initialize=0. This is to prevent accidental initializations. After adding tables, we need to:

UPDATE RPL.SUBSCRIPTION SET ACTIVE=1, INITIALIZE=1 WHERE SUBSCRIPTIONID = ?

Note 2: Frequency represents the minimum number of minutes between imports. This will work in coordination with the SQL agent job. For instance, if the job is set to run every minute, and subscription frequency is 5, then after a successful run that subscription will be skipped for the next 4 runs.

Note 3: A linked server to the publisher server is required only if you wish to use the automated data comparison (rpl.spSubscriptionCompare). If the network topology or security restrictions do not allow, we recommend to create a linked server from publisher to subscriber, so that the data compare can utilize the feature @RunAtPublisher =1, @Debug=1, @Exec=0 . This will print a script that can be executed at publisher.

1. For each table, replace the number 1 below with the SubscriptionId generated above:

exec rpl.spSubscribeTable 1, 'SchemaName', 'TableName'

Notice that the table must already exist in the subscriber. If it does not, use SSMS to generate a CREATE TABLE script from publisher, remove the columns you don’t need, then apply on subscriber. The table key must remain the same.

1. For custom processes that use special routines:

exec rpl.spSubscribeRoutine @SubscriptionId, @RoutineName, @Sequence

Note: the routine must be a stored proc and take these input parameters:

@subscriptionId int, @rvfrom varchar(20), @rvto varchar(20)

# Distribution

Replication is run by a .NET console app. So it can be executed anywhere. But here is our preferred method:

1. Copy contents of the folder:
2. \\MSync\ConsoleApp\MSync\bin\Debug

to distributer server c:\MSync

1. Grant the SQL Agent Service account DBO rights at both publisher and subscriber databases.
2. Create job with script: [Create Sql Agent Job.sql]

Note 1: The @command parameter has 3 components:

'C:\MSync "Data Source=.\S16;Initial Catalog=MyDb;Integrated Security=true;" "4"'

The red section must match the location used in (2), above.

The blue section represents the subscriber database and access credentials. For SQL authentication we must replace “Integrated Security=true” with “User Id=login;Password=password;”. Make sure the account in use has minimum permissions needed and only admins have access to that SQL instance or the login will be compromised.

The Purple section represents the number of threads.

For other optional parameters such as PriortyGroup, SubscriptionId and BatchSize please refer to the Architecture document.

# Adding Tables

Follow these steps to add tables:

exec rpl.spSubscribeTable SubscriptionId, 'SchemaName', 'TableName'

For considerations about whether to add a table to an existing subscription, or to create a new subscription, please refer to the architecture document.

# Removing Tables

Follow these steps to remove tables:

exec rpl.spUnPublishTable 'SchemaName', 'TableName'

exec rpl.spUnSubscribeTable 1, 'SchemaName', 'TableName'

This will drop the staging tables, delete triggers and replication procs, but will not remove the RV columns.

# Maintenance

In order to control the sizes of log tables, an SQL Agent Job performs these steps daily:

* Deletes records on RPL.DEL tables older than 30 days. This means if your subscription has not executed in 30 days you need to reinitialize.
* Deletes records rpl.ImportLogDetail older than 3 days, and rpl.ImportLog older than 7 days.
* Deletes records in rpl.Date tables older than 270 days. Between 90 and 270 days it keeps 1 record per day, and between 14 and 90 days it keeps 1 record per hour. Within 14 days it keeps 1 record per minute.
* Delete records from CMSServer.MSync\_Catalog.shopservice.tbl\_products\_audit older than 30 days

The job runs daily on CMSServer, and it’s called [Load DBA].

Another job [Replication Latency Check] checks all subscriptions every 30 minutes and reports subscriptions running behind.