LISTING 4-1 Log shipping configuration

```
-- Execute the following statements at the Primary to configure Log Shipping
-- for the database [PRIMARY].[WideWorldImporters],
-- The script needs to be run at the Primary in the context of the [msdb] database.
--------------------------------------------------------------------------------
-- Adding the Log Shipping configuration

-- ****** Begin: Script to be run at Primary: [PRIMARY] ******

DECLARE @LS_BackupJobId     AS uniqueidentifier
DECLARE @LS_PrimaryId     AS uniqueidentifier
DECLARE @SP_Add_RetCode     As int

EXEC @SP_Add_RetCode = master.dbo.sp_add_log_shipping_primary_database
        @database = N'WideWorldImporters'
        ,@backup_directory = N'\\STORAGE\Log_Shipping'
        ,@backup_share = N'\\STORAGE\Log_Shipping'
        ,@backup_job_name = N'[LOGSHIP] Log Backup WideWorldImporters'
        ,@backup_retention_period = 4320
        ,@backup_compression = 2
        ,@backup_threshold = 60
        ,@threshold_alert_enabled = 1
        ,@history_retention_period = 5760
        ,@backup_job_id = @LS_BackupJobId OUTPUT
        ,@primary_id = @LS_PrimaryId OUTPUT
        ,@overwrite = 1

IF (@@ERROR = 0 AND @SP_Add_RetCode = 0)
BEGIN

DECLARE @LS_BackUpScheduleUID   As uniqueidentifier
DECLARE @LS_BackUpScheduleID   AS int

EXEC msdb.dbo.sp_add_schedule
        @schedule_name =N'Every 15 minutes'
        ,@enabled = 1
        ,@freq_type = 4
        ,@freq_interval = 1
        ,@freq_subday_type = 4
        ,@freq_subday_interval = 15
        ,@freq_recurrence_factor = 0
        ,@active_start_date = 20170302     -- Change as appropriate
        ,@active_end_date = 99991231
        ,@active_start_time = 0
        ,@active_end_time = 235900
        ,@schedule_uid = @LS_BackUpScheduleUID OUTPUT
        ,@schedule_id = @LS_BackUpScheduleID OUTPUT
```

```
EXEC msdb.dbo.sp_attach_schedule
        @job_id = @LS_BackupJobId
        ,@schedule_id = @LS_BackUpScheduleID

EXEC msdb.dbo.sp_update_job
        @job_id = @LS_BackupJobId
        ,@enabled = 1

END

EXEC master.dbo.sp_add_log_shipping_alert_job

EXEC master.dbo.sp_add_log_shipping_primary_secondary
        @primary_database = N'WideWorldImporters'
        ,@secondary_server = N'SECONDARY'
        ,@secondary_database = N'WideWorldImporters'
        ,@overwrite = 1

-- ****** End: Script to be run at Primary: [PRIMARY]  ******

-- Execute the following statements at the Secondary to configure Log Shipping
-- for the database [SECONDARY].[WideWorldImporters],
-- the script needs to be run at the Secondary in the context of the [msdb] database.
-----------------------------------------------------------------------------------------
-- Adding the Log Shipping configuration

-- ****** Begin: Script to be run at Secondary: [SECONDARY] ******

DECLARE @LS_Secondary__CopyJobId  AS uniqueidentifier
DECLARE @LS_Secondary__RestoreJobId     AS uniqueidentifier
DECLARE @LS_Secondary__SecondaryId     AS uniqueidentifier
DECLARE @LS_Add_RetCode     As int

EXEC @LS_Add_RetCode = master.dbo.sp_add_log_shipping_secondary_primary
        @primary_server = N'PRIMARY'
        ,@primary_database = N'WideWorldImporters'
        ,@backup_source_directory = N'\\STORAGE\Log_Shipping'
        ,@backup_destination_directory = N'B:\PRIMARY_LOG_SHIPPING'
        ,@copy_job_name = N'[LOGSHIP] Copy PRIMARY WideWorldImporters'
        ,@restore_job_name = N'[LOGSHIP] Restore PRIMARY WideWorldImporters'
        ,@file_retention_period = 4320
        ,@overwrite = 1
        ,@copy_job_id = @LS_Secondary__CopyJobId OUTPUT
        ,@restore_job_id = @LS_Secondary__RestoreJobId OUTPUT
        ,@secondary_id = @LS_Secondary__SecondaryId OUTPUT

IF (@@ERROR = 0 AND @LS_Add_RetCode = 0)
BEGIN

DECLARE @LS_SecondaryCopyJobScheduleUID     As uniqueidentifier
DECLARE @LS_SecondaryCopyJobScheduleID     AS int

EXEC msdb.dbo.sp_add_schedule
        @schedule_name =N'Every 15 minutes'
```

```
        ,@enabled = 1
        ,@freq_type = 4
         ,@freq_interval = 1
        ,@freq_subday_type = 4
        ,@freq_subday_interval = 15
        ,@freq_recurrence_factor = 0
        ,@active_start_date = 20170302     -- Change as appropriate
        ,@active_end_date = 99991231
        ,@active_start_time = 0
        ,@active_end_time = 235900
        ,@schedule_uid = @LS_SecondaryCopyJobScheduleUID OUTPUT
        ,@schedule_id = @LS_SecondaryCopyJobScheduleID OUTPUT

EXEC msdb.dbo.sp_attach_schedule
        @job_id = @LS_Secondary__CopyJobId
        ,@schedule_id = @LS_SecondaryCopyJobScheduleID

DECLARE @LS_SecondaryRestoreJobScheduleUID     As uniqueidentifier
DECLARE @LS_SecondaryRestoreJobScheduleID     AS int

EXEC msdb.dbo.sp_add_schedule
        @schedule_name =N'Every 15 minutes'
        ,@enabled = 1
        ,@freq_type = 4
        ,@freq_interval = 1
        ,@freq_subday_type = 4
        ,@freq_subday_interval = 15
        ,@freq_recurrence_factor = 0
        ,@active_start_date = 20170302     -- Change as appropriate
        ,@active_end_date = 99991231
        ,@active_start_time = 0
        ,@active_end_time = 235900
        ,@schedule_uid = @LS_SecondaryRestoreJobScheduleUID OUTPUT
        ,@schedule_id = @LS_SecondaryRestoreJobScheduleID OUTPUT

EXEC msdb.dbo.sp_attach_schedule
        @job_id = @LS_Secondary__RestoreJobId
        ,@schedule_id = @LS_SecondaryRestoreJobScheduleID

END

DECLARE @LS_Add_RetCode2  As int

IF (@@ERROR = 0 AND @LS_Add_RetCode = 0)
BEGIN

EXEC @LS_Add_RetCode2 = master.dbo.sp_add_log_shipping_secondary_database
        @secondary_database = N'WideWorldImporters'
        ,@primary_server = N'PRIMARY'
        ,@primary_database = N'WideWorldImporters'
        ,@restore_delay = 0
        ,@restore_mode = 0
        ,@disconnect_users  = 0
        ,@restore_threshold = 45
        ,@threshold_alert_enabled = 1
        ,@history_retention_period   = 5760
        ,@overwrite = 1
```

```
END

IF (@@error = 0 AND @LS_Add_RetCode = 0)
BEGIN

EXEC msdb.dbo.sp_update_job
        @job_id = @LS_Secondary__CopyJobId
        ,@enabled = 1

EXEC msdb.dbo.sp_update_job
        @job_id = @LS_Secondary__RestoreJobId
        ,@enabled = 1

END
-- ****** End: Script to be run at Secondary: [SECONDARY] ******
GO
```

LISTING 4-2 Log shipping configuration.

```
-- ****** Begin: Script to be run at Monitor: [DBA] ******

EXEC msdb.dbo.sp_processlogshippingmonitorsecondary
        @mode = 1
        ,@secondary_server = N'SECONDARY'
        ,@secondary_database = N'WideWorldImporters'
        ,@secondary_id = N''
        ,@primary_server = N'PRIMARY'
        ,@primary_database = N'WideWorldImporters'
        ,@restore_threshold = 45
        ,@threshold_alert = 14420
        ,@threshold_alert_enabled = 1
        ,@history_retention_period    = 5760
        ,@monitor_server = N'DBA'
        ,@monitor_server_security_mode = 1
-- ****** End: Script to be run at Monitor: [DBA] ******
```

LISTING 4-3 Availability group configuration

```
--- YOU MUST EXECUTE THE FOLLOWING SCRIPT IN SQLCMD MODE.
:Connect REPLICA1
USE [master]
GO

CREATE ENDPOINT [Hadr_endpoint]
    AS TCP (LISTENER_PORT = 5022)
    FOR DATA_MIRRORING (ROLE = ALL, ENCRYPTION = REQUIRED ALGORITHM AES)
GO

IF (SELECT state FROM sys.endpoints WHERE name = N'Hadr_endpoint') <> 0
BEGIN
    ALTER ENDPOINT [Hadr_endpoint] STATE = STARTED
END
GO

use [master]
GO

GRANT CONNECT ON ENDPOINT::[Hadr_endpoint] TO [SQL\SQLServer]
GO

:Connect REPLICA1
IF EXISTS(SELECT * FROM sys.server_event_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER WITH (STARTUP_STATE=ON);
END
IF NOT EXISTS(SELECT * FROM sys.dm_xe_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER STATE=START;
END
GO

:Connect REPLICA2
USE [master]
GO

CREATE ENDPOINT [Hadr_endpoint]
    AS TCP (LISTENER_PORT = 5022)
    FOR DATA_MIRRORING (ROLE = ALL, ENCRYPTION = REQUIRED ALGORITHM AES)
GO

IF (SELECT state FROM sys.endpoints WHERE name = N'Hadr_endpoint') <> 0
BEGIN
    ALTER ENDPOINT [Hadr_endpoint] STATE = STARTED
END
GO

use [master]

GO

GRANT CONNECT ON ENDPOINT::[Hadr_endpoint] TO [SQL\SQLServer]
GO
```

```
:Connect REPLICA2
IF EXISTS(SELECT * FROM sys.server_event_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER WITH (STARTUP_STATE=ON);
END
IF NOT EXISTS(SELECT * FROM sys.dm_xe_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER STATE=START;
END
GO

:Connect REPLICA3
USE [master]
GO

CREATE ENDPOINT [Hadr_endpoint]
    AS TCP (LISTENER_PORT = 5022)
    FOR DATA_MIRRORING (ROLE = ALL, ENCRYPTION = REQUIRED ALGORITHM AES)
GO

IF (SELECT state FROM sys.endpoints WHERE name = N'Hadr_endpoint') <> 0
BEGIN
    ALTER ENDPOINT [Hadr_endpoint] STATE = STARTED
END
GO

use [master]
GO

GRANT CONNECT ON ENDPOINT::[Hadr_endpoint] TO [SQL\SQLServer]
GO

:Connect REPLICA3

IF EXISTS(SELECT * FROM sys.server_event_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER WITH (STARTUP_STATE=ON);
END
IF NOT EXISTS(SELECT * FROM sys.dm_xe_sessions WHERE name='AlwaysOn_health')
BEGIN
  ALTER EVENT SESSION [AlwaysOn_health] ON SERVER STATE=START;
END
GO

:Connect REPLICA1
USE [master]
GO

CREATE AVAILABILITY GROUP [WWI_AG]
WITH (AUTOMATED_BACKUP_PREFERENCE = SECONDARY,
```

```
DB_FAILOVER = OFF,
DTC_SUPPORT = NONE)
FOR DATABASE [WideWorldImporters]
REPLICA ON N'REPLICA1' WITH (ENDPOINT_URL = N'TCP://REPLICA1.SQL.LOCAL:5022', FAILOVER_
MODE = AUTOMATIC, AVAILABILITY_MODE = SYNCHRONOUS_COMMIT, BACKUP_PRIORITY = 50,
SECONDARY_ROLE(ALLOW_CONNECTIONS = READ_ONLY)),
    N'REPLICA2' WITH (ENDPOINT_URL = N'TCP://REPLICA2.SQL.LOCAL:5022', FAILOVER_MODE
= AUTOMATIC, AVAILABILITY_MODE = SYNCHRONOUS_COMMIT, BACKUP_PRIORITY = 75, SECONDARY_
ROLE(ALLOW_CONNECTIONS = READ_ONLY)),
    N'REPLICA3' WITH (ENDPOINT_URL = N'TCP://REPLICA3.SQL.LOCAL:5022', FAILOVER_MODE
= MANUAL, AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT, BACKUP_PRIORITY = 100, SECONDARY_
ROLE(ALLOW_CONNECTIONS = ALL));
GO

:Connect REPLICA1
USE [master]
GO

ALTER AVAILABILITY GROUP [WWI_AG]
ADD LISTENER N'WWI_LISTENER' (
WITH IP
((N'192.168.0.214', N'255.255.255.0')
)
, PORT=1433);
GO

:Connect REPLICA2
ALTER AVAILABILITY GROUP [WWI_AG] JOIN;
GO

:Connect REPLICA3
ALTER AVAILABILITY GROUP [WWI_AG] JOIN;
GO

:Connect REPLICA1
BACKUP DATABASE [WideWorldImporters] TO  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters.bak' WITH  COPY_ONLY, FORMAT, INIT, SKIP, REWIND, NOUNLOAD,
COMPRESSION,  STATS = 5
GO

:Connect REPLICA2
RESTORE DATABASE [WideWorldImporters] FROM  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters.bak' WITH  NORECOVERY,  NOUNLOAD,  STATS = 5
GO

:Connect REPLICA3
RESTORE DATABASE [WideWorldImporters] FROM  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters.bak' WITH  NORECOVERY,  NOUNLOAD,  STATS = 5
GO

:Connect REPLICA1
BACKUP LOG [WideWorldImporters] TO  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters_20170310165240.trn' WITH NOFORMAT, NOINIT, NOSKIP, REWIND, NOUNLOAD,
COMPRESSION,  STATS = 5
```

```
GO

:Connect REPLICA2
RESTORE LOG [WideWorldImporters] FROM  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters_20170310165240.trn' WITH  NORECOVERY,  NOUNLOAD,  STATS = 5
GO

:Connect REPLICA2
-- Wait for the replica to start communicating
begin try
declare @conn bit
declare @count int
declare @replica_id uniqueidentifier
declare @group_id uniqueidentifier
set @conn = 0
set @count = 30 -- wait for 5 minutes

if (serverproperty('IsHadrEnabled') = 1)
    and (isnull((select member_state from master.sys.dm_hadr_cluster_
members where upper(member_name COLLATE Latin1_General_CI_AS) =
upper(cast(serverproperty('ComputerNamePhysicalNetBIOS') as nvarchar(256)) COLLATE
Latin1_General_CI_AS)), 0) <> 0)
    and (isnull((select state from master.sys.database_mirroring_endpoints), 1) = 0)
begin
    select @group_id = ags.group_id from master.sys.availability_groups as ags where
name = N'WWI_AG'
    select @replica_id = replicas.replica_id from master.sys.availability_replicas
as replicas where upper(replicas.replica_server_name COLLATE Latin1_General_CI_AS) =
upper(@@SERVERNAME COLLATE Latin1_General_CI_AS) and group_id = @group_id
    while @conn <> 1 and @count > 0
    begin
        set @conn = isnull((select connected_state from master.sys.dm_hadr_availability_
replica_states as states where states.replica_id = @replica_id), 1)
        if @conn = 1
        begin
            -- exit loop when the replica is connected, or if the query cannot find the
replica status
            break
        end
        waitfor delay '00:00:10'
        set @count = @count - 1
    end
end
end try
begin catch
    -- If the wait loop fails, do not stop execution of the alter database statement
end catch
ALTER DATABASE [WideWorldImporters] SET HADR AVAILABILITY GROUP = [WWI_AG];
GO

:Connect REPLICA3
RESTORE LOG [WideWorldImporters] FROM  DISK = N'\\STORAGE\SQL_Backup\REPLICA1\
WideWorldImporters_20170310165240.trn' WITH  NORECOVERY,  NOUNLOAD,  STATS = 5
GO

:Connect REPLICA3
```

```
-- Wait for the replica to start communicating
begin try
declare @conn bit
declare @count int
declare @replica_id uniqueidentifier
declare @group_id uniqueidentifier
set @conn = 0
set @count = 30 -- wait for 5 minutes

if (serverproperty('IsHadrEnabled') = 1)
    and (isnull((select member_state from master.sys.dm_hadr_cluster_
members where upper(member_name COLLATE Latin1_General_CI_AS) =
upper(cast(serverproperty('ComputerNamePhysicalNetBIOS') as nvarchar(256)) COLLATE
Latin1_General_CI_AS)), 0) <> 0)
    and (isnull((select state from master.sys.database_mirroring_endpoints), 1) = 0)
begin
    select @group_id = ags.group_id from master.sys.availability_groups as ags where
name = N'WWI_AG'
    select @replica_id = replicas.replica_id from master.sys.availability_replicas
as replicas where upper(replicas.replica_server_name COLLATE Latin1_General_CI_AS) =
upper(@@SERVERNAME COLLATE Latin1_General_CI_AS) and group_id = @group_id
    while @conn <> 1 and @count > 0
    begin
        set @conn = isnull((select connected_state from master.sys.dm_hadr_availability_
replica_states as states where states.replica_id = @replica_id), 1)
        if @conn = 1
        begin
            -- exit loop when the replica is connected, or if the query cannot find the
replica status
            break
        end
        waitfor delay '00:00:10'
        set @count = @count - 1
    end
end
end try
begin catch
    -- If the wait loop fails, do not stop execution of the alter database statement
end catch
ALTER DATABASE [WideWorldImporters] SET HADR AVAILABILITY GROUP = [WWI_AG];
GO
```

LISTING 4-4 Read-only routing URL

```
-- Execute the following statements at the Primary to configure Log Shipping
ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA1' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://REPLICA1.SQL.LOCAL:1433'));
GO

ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA2' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://REPLICA2.SQL.LOCAL:1433'));
GO

ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA3' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://REPLICA3.SQL.LOCAL:1433'));
GO
```

LISTING 4-5 Read-only routing list

```
-- Execute the following statements at the Primary to configure Log Shipping
ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA1' WITH
(PRIMARY_ROLE (READ_ONLY_ROUTING_LIST=(N'REPLICA2',N'REPLICA3')));
GO

ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA2' WITH
(PRIMARY_ROLE (READ_ONLY_ROUTING_LIST=(N'REPLICA1',N'REPLICA3')));
GO

ALTER AVAILABILITY GROUP [WWI_AG]
MODIFY REPLICA ON
N'REPLICA3' WITH
(PRIMARY_ROLE (READ_ONLY_ROUTING_LIST=(N'REPLICA1',N'REPLICA2')));
```

LISTING 4-6 Configure load-balancing across read-only replicas

```
READ_ONLY_ROUTING_LIST = (('REPLICA1','REPLICA2','REPLICA3'), 'REPLICA4', 'REPLICA5')
```

LISTING 4-7 Manual fail over with no data loss

```
--- YOU MUST EXECUTE THE FOLLOWING SCRIPT IN SQLCMD MODE.

:Connect REPLICA2

ALTER AVAILABILITY GROUP [WWI_AG] FAILOVER;

GO
```

LISTING 4-8 Forced failover with potential data loss

```
--- YOU MUST EXECUTE THE FOLLOWING SCRIPT IN SQLCMD MODE.
:Connect REPLICA3

ALTER AVAILABILITY GROUP [WWI_AG] FORCE_FAILOVER_ALLOW_DATA_LOSS;
GO
```

LISTING 4-9 Creating an distributed Availability Group on the primary Availability Group

```
CREATE AVAILABILITY GROUP [DAG]
    WITH (DISTRIBUTED)
     AVAILABILITY GROUP ON
         'AG1' WITH  (
            LISTENER_URL = 'TCP://AG1_LISTENER:5022',
             AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
            FAILOVER_MODE = MANUAL,
            SEEDING_MODE = AUTOMATIC
        ),
        'AG2' WITH (
            LISTENER_URL = 'TCP://AG2-LISTENER:5022',
            AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
            FAILOVER_MODE = MANUAL,
            SEEDING_MODE = AUTOMATIC
        );
GO
```

LISTING 4-10 Joining a distributed Availability Group from the secondary Availability Group

```
ALTER AVAILABILITY GROUP [distributedag]
     JOIN
     AVAILABILITY GROUP ON
        'AG1' WITH (
            LISTENER_URL = 'tcp://ag1-listener:5022',
            AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
            FAILOVER_MODE = MANUAL,
            SEEDING_MODE = AUTOMATIC
        ),
         'AG2' WITH (
            LISTENER_URL = 'tcp://ag2-listener:5022',
            AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
             FAILOVER_MODE = MANUAL,
            SEEDING_MODE = AUTOMATIC
        );
GO
```