**SQL Server 2016 Always On Distributed Availability Groups(**Author: [Edwin Sarmiento](https://www.mssqltips.com/sqlserverauthor/18/edwin-sarmiento/))

<https://www.mssqltips.com/sqlservertip/5053/setup-and-implement-sql-server-2016-always-on-distributed-availability-groups/>

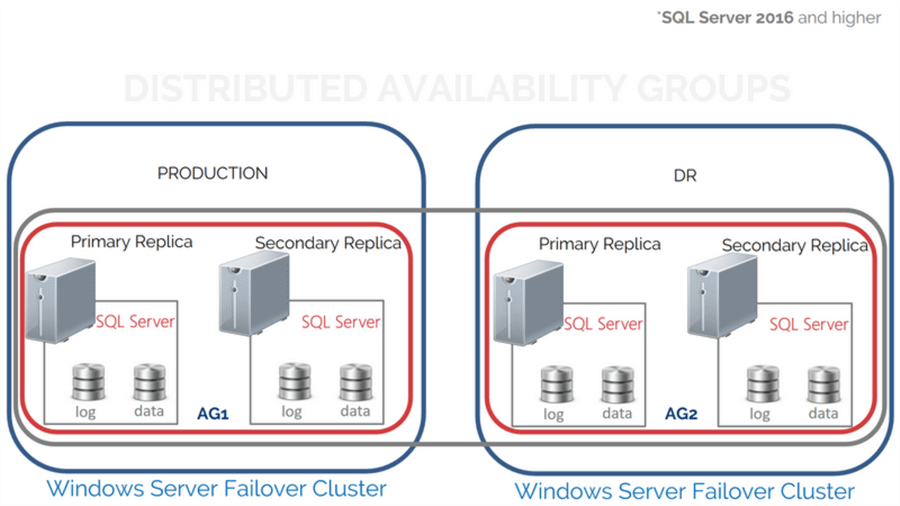
##### **Problem**

* We would like to implement SQL Server 2016 Always On Availability Groups for our disaster recovery strategy.
* We understand that SQL Server Always On Availability Groups require all replicas to be in the same Windows Server Failover Cluster.
* We already have an existing SQL Server Always On Availability Groups for local high availability.
* However, our internal server team has already built a separate Windows Server Failover Cluster in our disaster recovery site.
* How can we leverage SQL Server Always On Availability Group as a disaster recovery solution when our DR data center already has an existing Windows Server Failover Cluster?

##### **Solution**

* SQL Server 2016 introduced a new feature called Distributed Availability Group.
* A Distributed Availability Group is a special type of Availability Group that spans two separate Availability Groups.
* You can look at it as an “Availability Group of Availability Groups”.
* The underlying Availability Groups are configured on two different Windows Server Failover Clustering (WSFC) clusters. This makes it a viable solution for the scenario described in the problem statement.

A high-level diagram of a Distributed Availability Group is shown below.



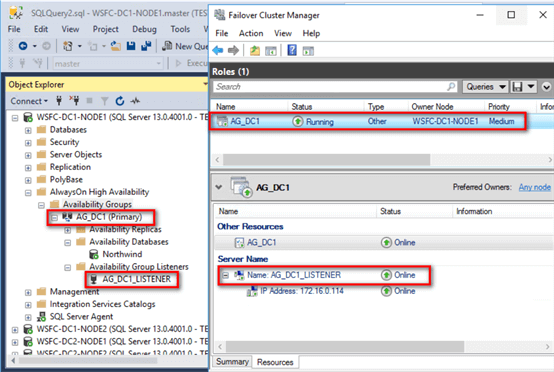
**Distributed Availability Groups solve a lot of challenges with the traditional Availability Groups.**

* **Disaster recovery solution for multi-site deployments**. In the past, you need to deploy a geographically stretched WSFC in order to have secondary replicas in a different data center for disaster recovery. This meant dealing with network configurations like the [CrossSubnetDelay](https://msdn.microsoft.com/en-us/library/jj151927(v=vs.85).aspx" \t "_blank) and [CrossSubnetThreshold](https://msdn.microsoft.com/en-us/library/jj151928(v=vs.85).aspx" \t "_blank) failover cluster properties as described in [this previous tip](https://www.mssqltips.com/sqlservertip/2533/sql-server-2012-multisubnet-cluster-part-2/) to avoid missed heartbeats that can potentially cause the failover cluster to go offline. It also meant choosing the appropriate quorum type and making sure that any issues with the failover cluster nodes on the DR data center does not affect the availability of the SQL Server Availability Group on the production data center. With Distributed Availability Groups, because you have two different WSFC, you treat them separately.
* **Operating System and/or hardware migration.**In previous versions of Windows Server, upgrading the operating system or the hardware meant provisioning a new WSFC, copying the configuration and databases, synchronizing them to minimize downtime during the upgrade and performing the cutover. You have to wait until after the OS or hardware migration is complete before you can create a new Availability Group. Your databases are at risk while you configure a high availability solution. And while Windows Server 2016 allows for Cluster OS migration, I still prefer a clean install on a new hardware or virtual machine for OS upgrades. Distributed Availability Groups allow you to create an Availability Group even before performing the cutover, giving your databases immediate high availability immediately after the cutover. You can have different versions of the WSFC operating system so long as you have the same version of SQL Server (2016 and higher only).
* **Scaling out readable secondary replicas.** Traditional Availability Groups allow for one primary and eight secondary replicas (from SQL Server 2014 and higher). With two Availability Groups, you get a total of eighteen (18) potential readable copies of the database – sixteen (16) secondary replicas, the primary replica of the first Availability Group and the primary replica of the second Availability Group. I don’t take this lightly because the licensing cost is not cheap. You have to really think about this before considering implementing Distributed Availability Groups mainly to scale out readable secondary replicas.

## Considerations with Distributed Availability Groups

There are several things to consider when deploying Distributed Availability Groups:

* **Metadata will not exist in the WSFC**. In addition to seeing the metadata in SQL Server via T-SQL, SSMS or PowerShell, traditional Availability Groups appear as cluster resources in the WSFC – the Availability Group is created as a resource group/role while the listener name is created as a virtual network name with a corresponding virtual IP address.  The screenshot below shows SSMS and the Failover Cluster Manager side-by-side displaying the Availability Group and listener name for a traditional Availability Group.

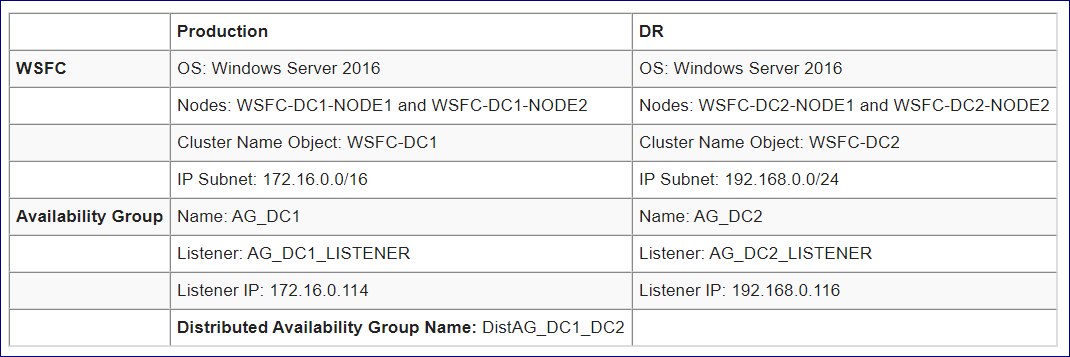


On the other hand, the metadata for Distributed Availability Groups only exist inside SQL Server. This means that administering a Distributed Availability group is solely the database administrator’s responsibility.

* **Listener name is required**. Unlike traditional Availability Groups where you can afford to not have a listener name and just use instance names for client application connectivity, Distributed Availability Groups require a listener name for each of the underlying Availability Group. The listener names are used as endpoints for the synchronization between the Availability Groups. What I don’t like about this is that you won’t be able to use dedicated network adapters for the Availability Group replication traffic. You have to use the network adapter that has connectivity to your DNS servers which is commonly the same as what the client applications use to connect to the database.
* **But the Distributed Availability Group does not have a listener name.**Only the underlying Availability Groups require a listener name; the Distributed Availability Group does not. This means read-only routing will not work with Distributed Availability Groups. You have to explicitly point the client applications to the instance names of the readable secondary replicas. If you are using this as a cluster OS upgrade strategy, you will need to rename the listener on the secondary Availability Group after the failover to use the listener on the original primary Availability Group.
* **Endpoint listeners should listen on all IP address.**For proper replication between the primary and secondary Availability Groups, the endpoints should be configured to listen to all IP addresses: LISTENER\_IP = ALL. Be sure to use this parameter when creating the endpoints. Otherwise, replication between the primary and secondary Availability Group will fail.
* **Only one read-write copy of the database**. Don’t be fooled – you can’t do load-balancing of read-write workloads. Just because you see two Availability Groups doesn’t mean you get two primary read-write replica databases. You only have one read-write copy of the database. The other primary replica on the secondary Availability Group functions similar to a distributor in a replication topology – it only receives transaction log records from the primary replica of the primary Availability Group and sends them to the other secondary replicas of the secondary Availability Group.
* **Efficient replication traffic between Availability Groups.**With traditional Availability Groups, one primary replica is responsible for sending all the transaction log records to all the secondary replicas. If, let’s say, you have four (4) secondary replicas in your DR data center, the primary replica will have to send the same set of transaction log records four (4) times across the network. As mentioned in the previous item, with Distributed Availability Groups, the primary replica of the primary Availability Group will only send the transaction log records to the primary replica of the secondary Availability Group. Transaction log records are only sent once instead of four (4) times.
* **Only manual failover is supported.**Because you have different WSFCs, there is no single coordinator that can handle automatic failover.
* **Currently limited to two (2) Availability Groups.** A Distributed Availability Group is limited to two Availability Groups. However, an Availability Group can be a member of more than one Distributed Availability Group. You can think of this as chaining Availability Groups.

#### **Step #1: Setup and Implement SQL Server 2016 Always On Distributed Availability Groups**

The prerequisites for deploying Distributed Availability Groups are no different from traditional Availability Groups. For the scenario described where the secondary Availability Group will be used as a DR solution, below are the details of the implementation.



**Here’s a high-level overview of the steps for your reference.**

1. Create the primary Availability Group (AG\_DC1) with a corresponding listener name (AG\_DC1\_LISTENER)
2. Create the Availability Group endpoint on all the replicas in the secondary Availability Group
3. Create login and grant the SQL Server service account CONNECT permissions to the endpoint
4. Create the secondary Availability Group (AG\_DC2) with a corresponding listener name (AG\_DC2\_LISTENER)
5. Join the secondary replicas to the secondary Availability Group
6. Create Distributed Availability Group (DistAG\_DC1\_DC2) on the primary Availability Group (AG\_DC1)
7. Join the secondary Availability Group (AG\_DC2) to the Distributed Availability Group

The primary Availability Group AG\_DC1 with the corresponding listener name has already been created. Refer to [this tip](https://www.mssqltips.com/sqlservertip/2518/sql-server-alwayson-availability-groups--part-2-availability-groups-setup/) on how to configure a traditional Availability Group.

#### **Step #2: Create Availability Group endpoint on all the replicas in the secondary Availability Group**

Use the T-SQL script below to create the endpoint on all of the replicas in the secondary Availability Group. The endpoints have already been created on the primary Availability Group as a side effect of Step #1. Be sure that the endpoint is listening on all IP addresses.

--Create endpoint on all Availability Group replicas

--Run this on the primary replica of the secondary Availability Group

:CONNECT WSFC-DC2-NODE1

USE [master]

GO

CREATE ENDPOINT [Hadr\_endpoint]

STATE=STARTED-

AS TCP (LISTENER\_PORT = 5022, LISTENER\_IP = ALL)

FOR DATA\_MIRRORING (ROLE = ALL, AUTHENTICATION = WINDOWS NEGOTIATE

, ENCRYPTION = REQUIRED ALGORITHM AES)

GO

--Run this on the secondary replica of the secondary Availability Group-

:CONNECT WSFC-DC2-NODE2

USE [master]

GO

CREATE ENDPOINT [Hadr\_endpoint]

STATE=STARTED

AS TCP (LISTENER\_PORT = 5022, LISTENER\_IP = ALL)

FOR DATA\_MIRRORING (ROLE = ALL, AUTHENTICATION = WINDOWS NEGOTIATE

, ENCRYPTION = REQUIRED ALGORITHM AES)

GO

#### **Step #3: Create login and grant the SQL Server service account CONNECT permissions to the endpoint**

Because the SQL Server service account will impersonate the SQL Server instance when connecting to the replicas – from primary to secondary Availability group and vice versa – you need to create it as a SQL Server login and grant it the CONNECT permissions to the endpoint. The same SQL Server service account – TESTDOMAIN\sqlservice - is used for all of the Availability Group replicas. Use the T-SQL script below to create the SQL Server service account as a login and grant it CONNECT permissions on the endpoint. Do this on all of the replicas of the secondary Availability Group.

--Create login and grant CONNECT permissions to the SQL Server service account

USE master

GO

CREATE LOGIN [TESTDOMAIN\sqlservice] FROM WINDOWS;

GO

GRANT CONNECT ON ENDPOINT::Hadr\_endpoint

TO [TESTDOMAIN\sqlservice];

GO

Note that if you are using a different SQL Server service account on each instance in the WSFC, you will need to create those as logins on the replicas that they need to connect to as well as grant CONNECT permissions to the endpoints. It becomes a bit more challenging to manage those service accounts. Even more complicated if the two WSFCs are in different Active Directory domains or no Active Directory domain at all. You will need to use certificates as described in [this tip](https://www.mssqltips.com/sqlservertip/4991/implement-a-sql-server-2016-availability-group-without-active-directory-part-1/).

#### **Step #4: Create the secondary Availability Group with a corresponding listener name**

Use the T-SQL script below to create the secondary Availability Group AG\_DC2. Be sure you are connected to the SQL Server instance that you want to configure as primary replica of the secondary Availability Group.

--Create second availability group on second failover cluster with replicas and listener

--Run this on the primary replica of the secondary Availability Group

:CONNECT WSFC-DC2-NODE1

CREATE AVAILABILITY GROUP [AG\_DC2]

FOR

REPLICA ON

N'WSFC-DC2-NODE1' WITH

(

ENDPOINT\_URL = N'TCP://WSFC-DC2-NODE1.TESTDOMAIN.COM:5022',

FAILOVER\_MODE = AUTOMATIC,

AVAILABILITY\_MODE = SYNCHRONOUS\_COMMIT,

SECONDARY\_ROLE(ALLOW\_CONNECTIONS = NO),

SEEDING\_MODE = AUTOMATIC

),

N'WSFC-DC2-NODE2' WITH

( ENDPOINT\_URL = N'TCP://WSFC-DC2-NODE2.TESTDOMAIN.COM:5022',

FAILOVER\_MODE = AUTOMATIC,

AVAILABILITY\_MODE = SYNCHRONOUS\_COMMIT,

SECONDARY\_ROLE(ALLOW\_CONNECTIONS = NO),

SEEDING\_MODE = AUTOMATIC

)

LISTENER 'AG\_DC2\_LISTENER'

(

WITH IP ( ('192.168.0.116','255.255.255.0') ) ,

PORT = 143

);

GO

The following parameters and their corresponding values are used to create the Availability Group

* **REPLICA**: Notice that I didn’t specify any database when I created the Availability Group.  Addition of the database in the Availability Group is done thru the next parameter
* **SEEDING\_MODE = AUTOMATIC**: a new parameter in SQL Server 2016 that introduces direct seeding, this allows creation of a database inside an Availability Group without performing the usual backup-copy-restore
* **FAILOVER\_MODE = AUTOMATIC:**specifies automatic failover of the Availability Group; AUTOMATIC since this is within the Availability Group itself, not the Distributed Availability Group
* **AVAILABILITY\_MODE = SYNCHRONOUS\_COMMIT:** specifies synchronous mode Availability Group replication; again, SYNCHRONOUS\_COMMIT since this is within the Availability Group itself, not the Distributed Availability Group
* **SECONDARY\_ROLE (ALLOW\_CONNECTIONS = NO):** specifies secondary replica databases to only be on standby and not used for read-only workloads

#### **Step #5: Join the secondary replicas to the secondary Availability Group**

The previous step simply created the Availability Group and defined the replicas. You need to join all the secondary replicas to the Availability Group.

--Join the secondary replicas to the secondary Availability Group

--Run this on the secondary replicas of the secondary Availability Group

:CONNECT WSFC-DC2-NODE2

ALTER AVAILABILITY GROUP [AG\_DC2] JOIN

--Allow the Availability Group to create databases on behalf of the primary replica

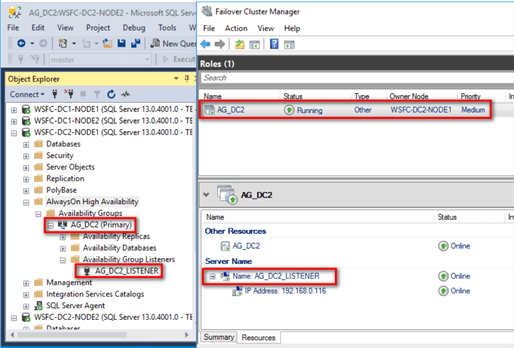
ALTER AVAILABILITY GROUP [AG\_DC2] GRANT CREATE ANY DATABASE

GO

The second statement – ALTER AVAILABILITY GROUP [agname] GRANT CREATE ANY DATABASE – is added because direct seeding was used in the creation of the Availability Group. Should you decide to manually perform the backup-copy-restore process to initialize the database, you can use this command instead:

ALTER DATABASE [dbName] SET HADR AVAILABILITY GROUP = [agName]

Now, because this is still a traditional Availability Group, you should see it as a resource group/role from within the Failover Cluster Manager.



Don’t get over-excited just yet. Because you don’t have the Distributed Availability Group yet, the secondary replica on the secondary Availability Group will report as NOT SYNCHRONIZING. That’s because you don’t have any database yet in the Availability Group. A database will be added when you create the Distributed Availability Group.

#### **Step #6: Create Distributed Availability Group on the primary Availability Group**

* Once the secondary Availability Group has been created, you can now proceed to create the Distributed Availability Group.
* Use the T-SQL script below to create the Distributed Availability Group DistAG\_DC1\_DC2.
* Be sure you are connected to the SQL Server instance that you want to configure as primary replica of the primary Availability Group.

--Create Distributed Availability Group

--Run this on the primary replica of the primary Availability Group

:CONNECT WSFC-DC1-NODE1

CREATE AVAILABILITY GROUP [DistAG\_DC1\_DC2]

WITH (DISTRIBUTED)

AVAILABILITY GROUP ON

'AG\_DC1' WITH

(

LISTENER\_URL = 'TCP://AG\_DC1\_LISTENER.TESTDOMAIN.COM:5022',

AVAILABILITY\_MODE = ASYNCHRONOUS\_COMMIT,

FAILOVER\_MODE = MANUAL,

SEEDING\_MODE = AUTOMATIC

),

'AG\_DC2' WITH

(

LISTENER\_URL = 'TCP://AG\_DC2\_LISTENER.TESTDOMAIN.COM:5022',

AVAILABILITY\_MODE = ASYNCHRONOUS\_COMMIT,

FAILOVER\_MODE = MANUAL,

SEEDING\_MODE = AUTOMATIC

);

GO

The following parameters and their corresponding values are used to create the Availability Group

* **DISTRIBUTED**: This tells SQL Server that you are creating a Distributed Availability Group
* **LISTENER\_URL**: Notice that this parameter was used instead of ENDPOINT\_URL. This parameter specifies the listener for each Availability Group along with the endpoint of the Availability Group – 5022 - not the endpoint of the listener - 1433.
* **FAILOVER\_MODE = MANUAL:**specifies automatic failover of the Availability Group; MANUAL since this now the Distributed Availability Group
* **AVAILABILITY\_MODE = ASYNCHRONOUS\_COMMIT:** specifies synchronous mode Availability Group replication; ASYNCHRONOUS\_COMMIT since this is now the Distributed Availability Group

Be careful with specifying the LISTENER\_URL parameter. When the listener was created, a corresponding Active Directory virtual computer object and a DNS entry are created. I used port 1433 as the listener port number – of course, with a different IP address - to simplify things since this is the same port number as the default SQL Server instance and will likely be opened in your network firewall. However, port 5022 is used to connect to the endpoint for log record synchronization. Since accessing a network resource requires an IP address and a port number, you can think of this as using the listener to find the secondary replica and port 5022 to connect to the endpoint. Be sure to open up port 5022 in your network firewall.

#### **Step #7: Join the secondary Availability Group to the Distributed Availability Group**

Once the Distributed Availability Group has been created, you can now proceed to join the secondary Availability Group. Use the T-SQL script below to join the Availability Group AD\_DC2 to the Distributed Availability Group DistAG\_DC1\_DC2. Be sure you are connected to the SQL Server instance that you want to configure as primary replica of the secondary Availability Group.

--Create second availability group on second failover cluster with replicas and listener

--Run this on the primary replica of the secondary Availability Group

:CONNECT WSFC-DC2-NODE1

ALTER AVAILABILITY GROUP [DistAG\_DC1\_DC2]

JOIN

AVAILABILITY GROUP ON

'AG\_DC1' WITH

(

LISTENER\_URL = 'TCP://AG\_DC1\_LISTENER.TESTDOMAIN.COM:5022',

AVAILABILITY\_MODE = ASYNCHRONOUS\_COMMIT,

FAILOVER\_MODE = MANUAL,

SEEDING\_MODE = AUTOMATIC

),

'AG\_DC2' WITH

(

LISTENER\_URL = 'TCP://AG\_DC2\_LISTENER.TESTDOMAIN.COM:5022',

AVAILABILITY\_MODE = ASYNCHRONOUS\_COMMIT,

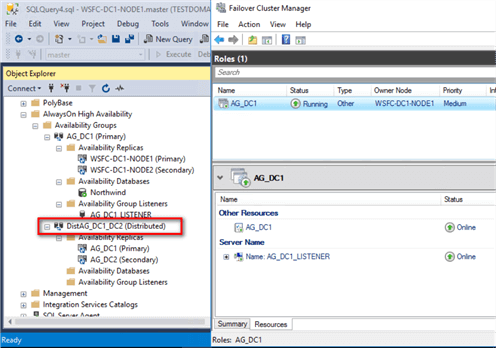
FAILOVER\_MODE = MANUAL,

SEEDING\_MODE = AUTOMATIC

);

GO

* Review the Distributed Availability Group by expanding the Availability Groups folder in SSMS.
* Notice the word Distributed appended to the Distributed Availability Group.



* Also, notice that there is no resource group/role created in any of the WSFC.
* As mentioned, the metadata is all stored within SQL Server. In fact, even SSMS does not display the name of the databases in the Distributed Availability Group.
* You can use the T-SQL script below to view the metadata and status of the Distributed Availability Group.

**--View metadata and status of the Distributed Availability Group**

SELECT r.replica\_server\_name, r.endpoint\_url,

rs.connected\_state\_desc, rs.role\_desc, rs.operational\_state\_desc,

rs.recovery\_health\_desc,rs.synchronization\_health\_desc,

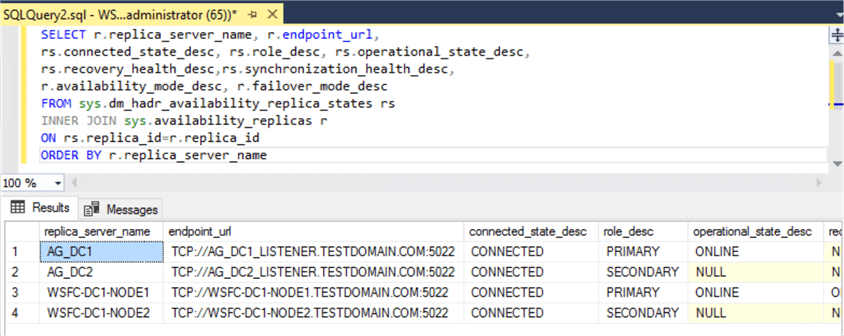
r.availability\_mode\_desc, r.failover\_mode\_desc

FROM sys.dm\_hadr\_availability\_replica\_states rs

INNER JOIN sys.availability\_replicas r

ON rs.replica\_id=r.replica\_id

ORDER BY r.replica\_server\_name



* Implementing Distributed Availability Groups requires proper planning and thorough documentation.
* It’s not as simple as it is. Use this tip as a guide to help you successfully provide a disaster recovery solution for your SQL Server databases using Distributed Availability Group.