



MS SQL Server *Distributed* AG on OCI

Architecture example

Deployment Document

10th of December 2024 | Version 1.0

Copyright © 2024, Oracle and/or its affiliates

Contents

Document Control	3
1.1 Version Control	3
1.2 Team	3
1.3 Document Purpose.....	3
MS SQL Server <i>Distributed</i> Availability Group on OCI	4
1.4 Logical Representation:	5
1.5 Distributed AOAG creation:	10
1.6 Failover procedure:	12
1.6.1 Step 0	12
1.6.2 Step 1	12
1.6.3 Step 2.....	13
1.6.4 Step 3	13
1.6.5 Step 4	14
1.6.6 Step 5	16
1.7 Fallback procedure	16

Document Control

1.1 Version Control

Version	Authors	Date	Comments
1.0	Alessandro Volpi	10 December 2024	First version

1.2 Team

Name	Email	Role	Company
Alessandro Volpi	alessandro.volpi@oracle.com	Cloud Solution Specialist	Oracle

1.3 Document Purpose

This document provides some guidance to create a Microsoft SQL Server Distributed Always On Availability Group Solution on OCI.

MS SQL Server *Distributed Availability Group* on OCI

A Distributed Availability Group is a powerful feature in Microsoft SQL Server that extends the capabilities of traditional Availability Group for SQL Server. You already know you are able to run SQL Server Availability Group on OCI

Deploy Microsoft SQL Server Always On Availability Groups for HA and DR on OCI:

<https://docs.oracle.com/en/learn/oci-sql-server-aoag/index.html>

and Distributed AOAG allows you to create a Disaster Recovery solution that spans multiple Windows Server Failover Clusters (WSFCs) running in different OCI region. This enables you to achieve higher levels of availability, disaster recovery, and geographic distribution for your critical SQL Server databases running on OCI.

The reference documentation is the official Microsoft published here:

What is a distributed availability group - SQL Server Always On | Microsoft Learn:

<https://learn.microsoft.com/en-us/sql/database-engine/availability-groups/windows/distributed-availability-groups?view=sql-server-ver16>

Configure a distributed availability group - SQL Server Always On | Microsoft Learn:

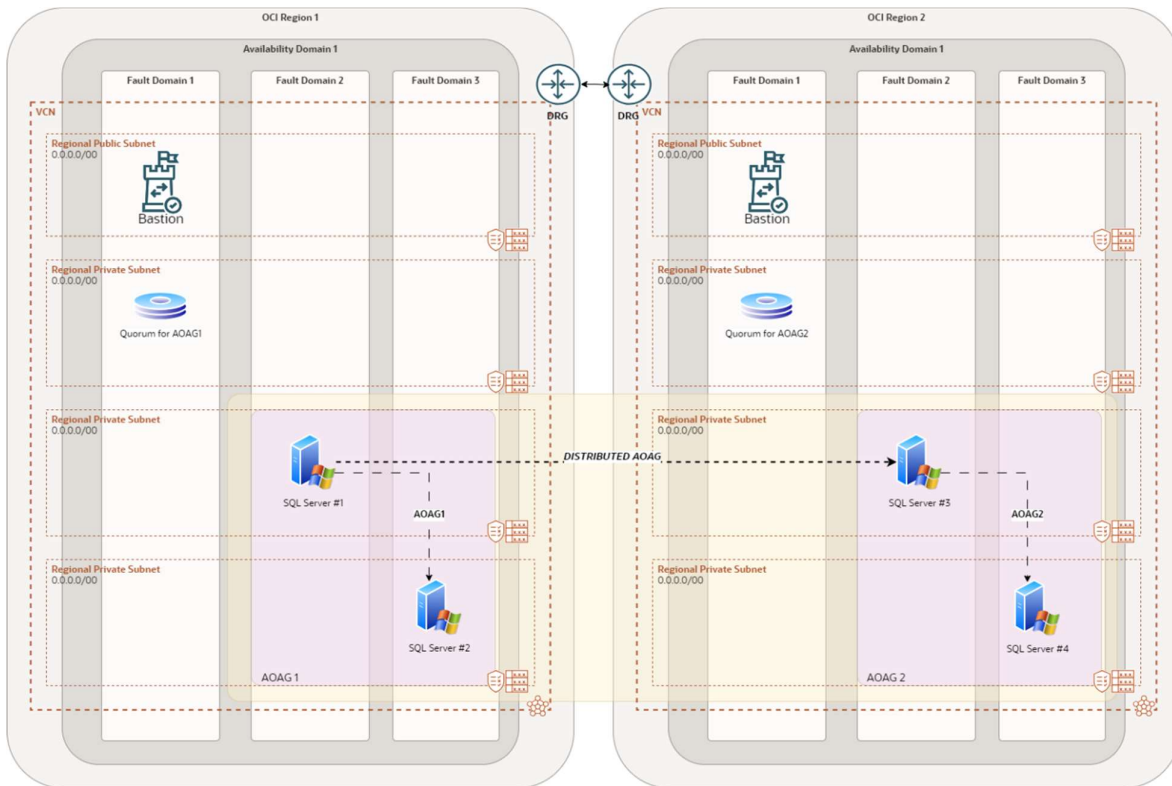
<https://learn.microsoft.com/en-us/sql/database-engine/availability-groups/windows/configure-distributed-availability-groups?view=sql-server-ver15&tabs=automatic%2Csql22>

The building blocks of a Distributed AOAG are:

- OCI VCNs created into 2 separate OCI regions and connected through DRGs remote peering.
- AOAG #1 running into OCI Region: this is where the Database to be replicated runs normally. This is based on a Windows Server Failover Cluster running on Server #1 and Server #2 into following example.
- AOAG #2 running into OCI Region 2: this is completely independent AOAG running on a WSFC composed by Server #3 and Server #4 into following example.
- Distributed AOAG: this is a logical construct created into the SQL Database to be replicated.

1.4 Logical Representation:

Here a logical representation of a Distributed AOAG:



In this document we will not cover detailed instructions on how create the 2 independent AOAG (one in the first region and the other into the second region) because they could be found here:

<https://docs.oracle.com/en/learn/oci-sql-server-aoag/index.html>

So let's assume we already have 2 independent AOAG running in 2 different OCI regions peered, in this example the OCI regions are Paris and Marseille.

Here we have *paris-wsfc* that is the first WSFC cluster into the first region with *paris-aoag* that is the first SQL AOAG and *paris-sql-list* is the SQL Listener for the SQL AOAG. The 2 Windows nodes are *sql-srv1* and *sql-srv2*

The screenshot displays the Failover Cluster Manager interface. The left-hand tree view shows the hierarchy: Failover Cluster Manager > paris-wsfc.acme.corp > Roles. The main pane is titled 'Roles (1)' and contains a table with the following data:

Name	Status	Type	Owner Node	Priority	Information
paris-aoag	Running	Other	sql-srv1	Medium	

Below the 'Roles (1)' section, there is a detailed view for the 'paris-aoag' role. It includes a table for 'Other Resources' and a section for 'Server Name' with associated IP addresses and their status.

Name	Status	Information
paris-aoag	Online	

Server Name

Name	Status	Information
paris-sql-list	Online	
IP Address: 172.16.2.23	Online	
IP Address: 172.16.3.23	Offline	

In the second region we have *marseille-wsfc* that is the second WSFC cluster with *marseille-aoag* that is the second SQL AOAG and *mars-sql-list* that is the SQL Listener for the second SQL AOAG. The 2 Windows nodes are *sql-srv3* and *sql-srv4*

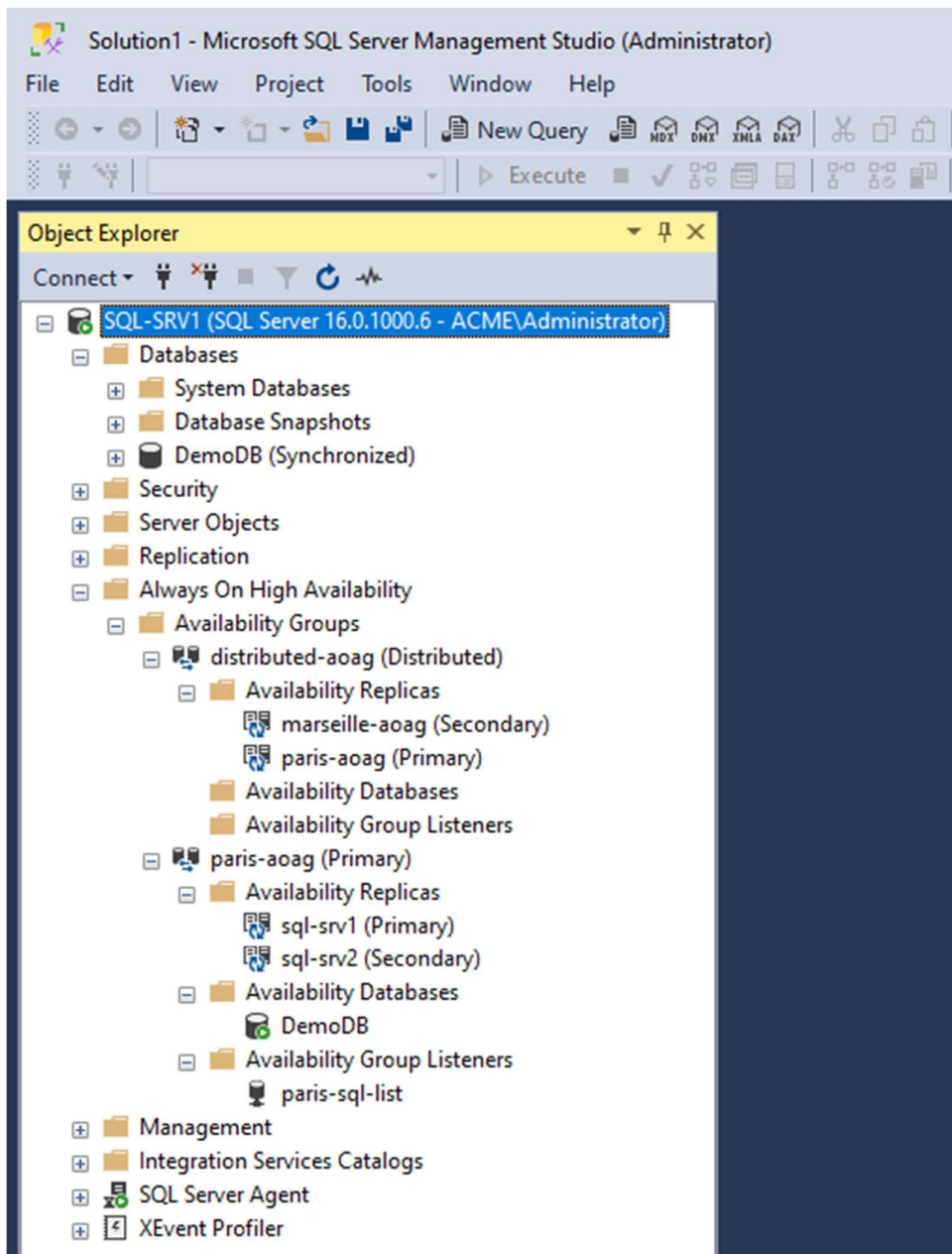
The screenshot displays the Failover Cluster Manager console. The left-hand tree view shows the hierarchy: Failover Cluster Manager > marseille-wsfc.acme.corp > Roles. The main pane is titled 'Roles (1)' and contains a table with the following data:

Name	Status	Type	Owner Node	Priority
marseille-aoag	Running	Other	sql-srv3	Medium

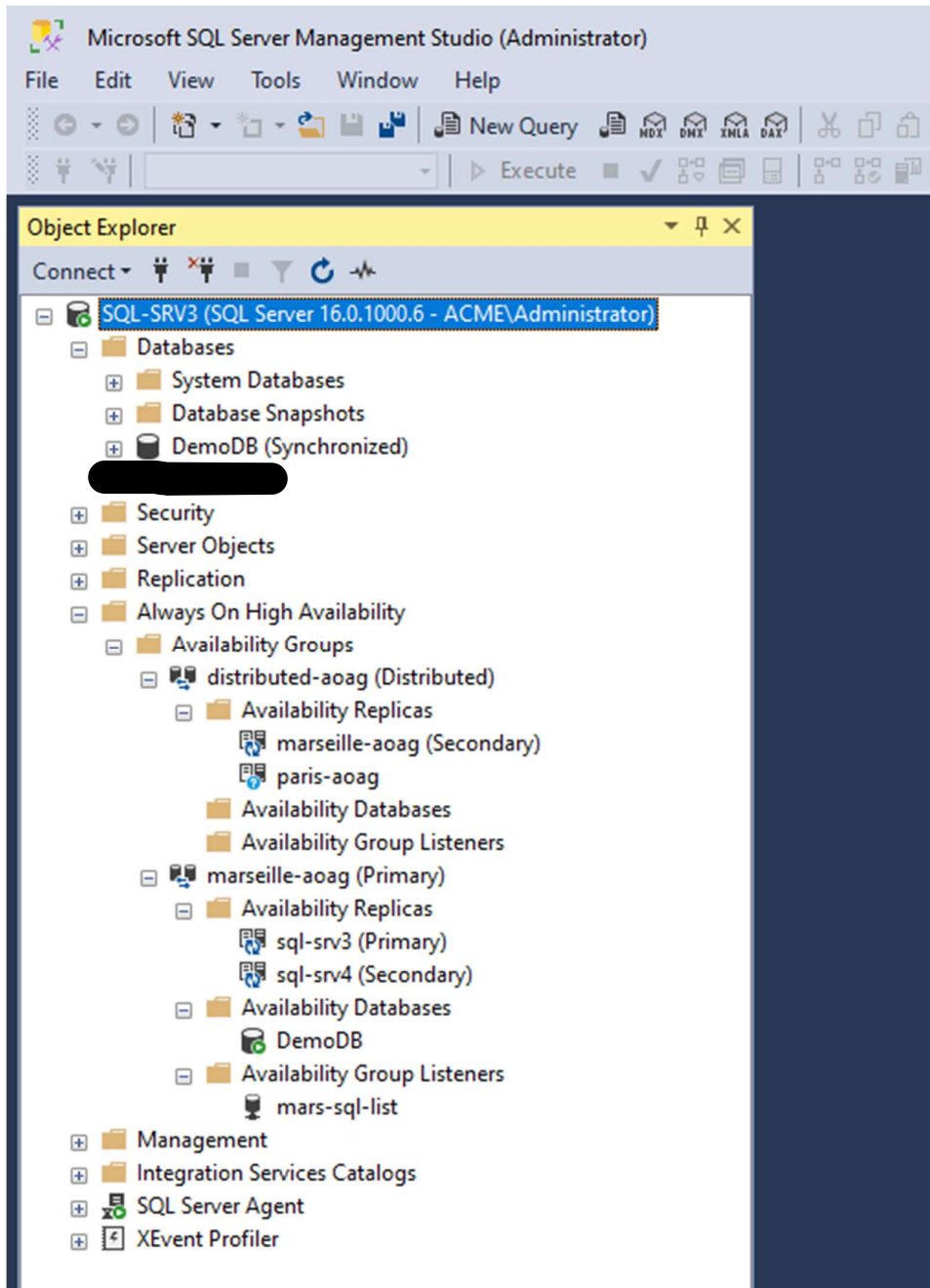
Below this, the 'marseille-aoag' role is expanded, showing its configuration details:

Name	Status	Information
Other Resources		
marseille-aoag	Online	
Server Name		
Name: mars-sql-list	Online	
IP Address: 172.17.2.23	Online	
IP Address: 172.17.3.23	Offline	

From a SQL Server perspective, starting from *sql-srv1 (paris-aoag)* we can see in this example the “*DemoDB*” that is the database replicated with the first AOAG and the newly created *distributed-aoag*



So connecting with *sql-srv3 (marseille-aoag)* we can see in this example also the “*DemoDB*” that is the database replicated with the first AOAG, the newly created *distributed-aoag*, and the *marseille-aoag* that is the second AOAG created into the second site (Marseille)



1.5 Distributed AOAG creation:

Now let's see how to create the “distributed-aoag” composed by the 2 underlying already running AOAG.

As already mentioned, we assume that 2 independent AOAG are already up and running in 2 different OCI region.

Please note that the second AOAG, the “standby” one, in this case *marseille-aoag*, needs to have no databases associated, so practically the second AOAG needs to be “empty” before the distributed AOAG creation, so without any availability database associated. You can create the second AOAG as usual with an initial DB associated and after that you can remove this database that has been used only to create the second AOAG. This because with the graphical interface is not possible to create an AOAG without any database associated.

Let's start creating the distributed AOAG on the first AOAG, so connecting to SQL Server on the first server (PARIS site sql-srv1 node), execute:

Please pay attention to:

- the listener names (**paris-sql-list**, **mars-sql-list**)
- TCP port to be used, **5022** is the port of the ENDPOINT, and must be used. This is usually different from the Listener PORT (1433)
- Availability group names, must be exactly the names used by the already running AOAG

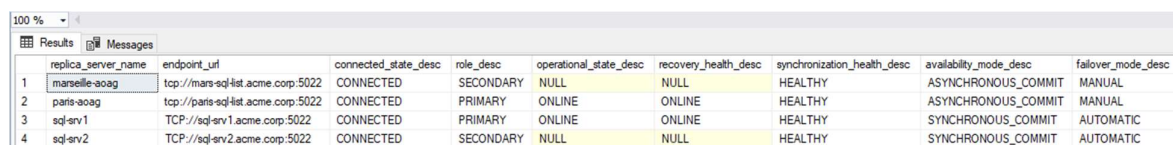
```
USE MASTER;
CREATE AVAILABILITY GROUP [distributed-aoag]
WITH (DISTRIBUTED)
AVAILABILITY GROUP ON
    'paris-aoag' WITH
    (
        LISTENER_URL = 'tcp://paris-sql-list.acme.corp:5022',
        AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
        FAILOVER_MODE = MANUAL,
        SEEDING_MODE = AUTOMATIC
    ),
    'marseille-aoag' WITH
    (
        LISTENER_URL = 'tcp://mars-sql-list.acme.corp:5022',
        AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
        FAILOVER_MODE = MANUAL,
        SEEDING_MODE = AUTOMATIC
    );
GO
```

Now join to the distributed AOAG on the second AOAG, so connecting to SQL Server on the first server of the second AOAG (MARSEILLE site sql-srv3 server), execute:

```
USE MASTER;
ALTER AVAILABILITY GROUP [distributed-aoag]
JOIN
AVAILABILITY GROUP ON
    'paris-aoag' WITH
    (
        LISTENER_URL = 'tcp://paris-sql-list.acme.corp:5022',
        AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
        FAILOVER_MODE = MANUAL,
        SEEDING_MODE = AUTOMATIC
    ),
    'marseille-aoag' WITH
    (
        LISTENER_URL = 'tcp://mars-sql-list.acme.corp:5022',
        AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
        FAILOVER_MODE = MANUAL,
        SEEDING_MODE = AUTOMATIC
    );
GO
```

It's important to understand that unlike traditional Availability Group, Distributed Availability Groups don't require resource groups or roles in the WSFC. All metadata is managed within SQL Server. This means that even SQL Server Management Studio doesn't directly display the names of databases in the Distributed Availability Group. To view this information, execute the following T-SQL script.

```
--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
```



	replica_server_name	endpoint_url	connected_state_desc	role_desc	operational_state_desc	recovery_health_desc	synchronization_health_desc	availability_mode_desc	failover_mode_desc
1	marseille-aoag	tcp://mars-sql-list.acme.corp:5022	CONNECTED	SECONDARY	NULL	NULL	HEALTHY	ASYNCHRONOUS_COMMIT	MANUAL
2	paris-aoag	tcp://paris-sql-list.acme.corp:5022	CONNECTED	PRIMARY	ONLINE	ONLINE	HEALTHY	ASYNCHRONOUS_COMMIT	MANUAL
3	sql-srv1	TCP://sql-srv1.acme.corp:5022	CONNECTED	PRIMARY	ONLINE	ONLINE	HEALTHY	SYNCHRONOUS_COMMIT	AUTOMATIC
4	sql-srv2	TCP://sql-srv2.acme.corp:5022	CONNECTED	SECONDARY	NULL	NULL	HEALTHY	SYNCHRONOUS_COMMIT	AUTOMATIC

Please note that to accommodate potential network latency between regions, we've configured the primary and secondary AOAG with ASYNCHRONOUS COMMIT replication. This minimizes performance overhead on the primary database. Within each AOAG, we've opted for SYNCHRONOUS COMMIT replication among replicas to ensure high availability. However, for failover between ASYNCHRONOUS COMMIT replicas (in case case of Distributed AOAG), reducing data loss requires a temporary switch to SYNCHRONOUS COMMIT mode before initiating the failover. For the FAILOVER_MODE the only available mode for Distributed AG is MANUAL.

1.6 Failover procedure:

Now let's talk about FAILOVER between the 2 AOAG.

Failover procedure of the Database is composed by some easy steps and checks:

- Initial checks
- Change the AVAILABILITY MODE (asynchronous to synchronous) for the Primary AOAG and for the Secondary AOAG
- Run scripts to check if the synchronization is fine
- Change the role of the Primary AOAG (primary to secondary)
- Failover to the secondary AOAG
- Change the AVAILABILITY MODE (synchronous to asynchronous) for the Primary AOAG and for the Secondary AOAG

1.6.1 Step 0

Run this scripts to check if synchronization is fine:

```
select ag.name, ag.is_distributed, ar.replica_server_name, ar.availability_mode_desc,
ars.connected_state_desc, ars.role_desc,
    ars.operational_state_desc, ars.synchronization_health_desc from sys.availability_groups
ag
    join sys.availability_replicas ar on ag.group_id=ar.group_id
    left join sys.dm_hadr_availability_replica_states ars
    on ars.replica_id=ar.replica_id
where ag.is_distributed=1
GO
```

	name	is_distributed	replica_server_name	availability_mode_desc	connected_state_desc	role_desc	operational_state_desc	synchronization_health_desc
1	distributed-aoag	1	paris-aoag	SYNCHRONOUS_COMMIT	CONNECTED	PRIMARY	ONLINE	HEALTHY
2	distributed-aoag	1	marseille-aoag	SYNCHRONOUS_COMMIT	CONNECTED	SECONDARY	NULL	HEALTHY

	name	is_distributed	replica_server_name	availability_mode_desc	connected_state_desc	role_desc	operational_state_desc	synchronization_health_desc
1	distributed-aoag	1	paris-aoag	SYNCHRONOUS_COMMIT	NULL	NULL	NULL	NULL
2	distributed-aoag	1	marseille-aoag	SYNCHRONOUS_COMMIT	CONNECTED	SECONDARY	ONLINE	HEALTHY

1.6.2 Step 1

Change the Availability Mode, execute this script first on the currently Primary SQL of the currently Primary site and then on the Primary SQL of the Secondary site:

```
--Run this first on the primary replica of the primary AOAG and then again on the secondary
AOAG
USE MASTER;
ALTER AVAILABILITY GROUP [distributed-aoag]
MODIFY
AVAILABILITY GROUP ON
'paris-aoag' WITH
(
    AVAILABILITY_MODE = SYNCHRONOUS_COMMIT
),
'marseille-aoag' WITH
(
    AVAILABILITY_MODE = SYNCHRONOUS_COMMIT
);
```

Step 2 run this scripts to check if synchronization is fine:

```

select ag.name, ag.is_distributed, ar.replica_server_name, ar.availability_mode_desc,
ars.connected_state_desc, ars.role_desc,
    ars.operational_state_desc, ars.synchronization_health_desc from sys.availability_groups
ag
join sys.availability_replicas ar on ag.group_id=ar.group_id
left join sys.dm_hadr_availability_replica_states ars
on ars.replica_id=ar.replica_id
where ag.is_distributed=1
GO

```

1.6.3 Step 2

To avoid any data loss please be sure to check this step results, the status needs to be SYNCHRONIZED, and the last_hardened_lsn needs to be the same per database on both the global primary and forwarder.

```

-- Run this query on the Global Primary and the forwarder
-- Check the results to see if synchronization_state_desc is SYNCHRONIZED, and the
last_hardened_lsn is the same per database on both the global primary and forwarder
-- If not rerun the query on both side every 5 seconds until it is the case
--

```

```

SELECT ag.name
    , drs.database_id
    , db_name(drs.database_id) as database_name
    , drs.group_id
    , drs.replica_id
    , drs.synchronization_state_desc
    , drs.last_hardened_lsn
FROM sys.dm_hadr_database_replica_states drs
INNER JOIN sys.availability_groups ag on drs.group_id = ag.group_id;

```

	name	database_id	database_name	group_id	replica_id	synchronization_state_desc	last_hardened_lsn
1	paris-aoag	5	DemoDB	0C4C1F27-9556-42CB-BB38-010DE188835D	D7588BEF-2491-4AD4-889B-A697495D4996	SYNCHRONIZED	41000000062400001
2	paris-aoag	5	DemoDB	0C4C1F27-9556-42CB-BB38-010DE188835D	BBE95F61-C162-4AB7-8A13-FDFB22C077ED	SYNCHRONIZED	41000000062400001
3	distributed-aoag	5	DemoDB	F6D2F5BD-20D8-E006-04FB-1A36FEEAABF6	6090AB43-14A5-F8A1-A0AB-242CD25F8761	SYNCHRONIZED	41000000062400001

	name	database_id	database_name	group_id	replica_id	synchronization_state_desc	last_hardened_lsn
1	marseille-aoag	6	DemoDB	08AA8CFA-C63C-42A2-A143-C05AC5A57359	246F8FC3-4F72-498F-9A83-2BE8DF760AEC	SYNCHRONIZED	41000000062400001
2	marseille-aoag	6	DemoDB	08AA8CFA-C63C-42A2-A143-C05AC5A57359	C704E755-E998-4FF1-A880-790381C5101B	SYNCHRONIZED	41000000062400001

1.6.4 Step 3

Now on the currently Primary SQL of the currently Primary site you are ready to set
REQUIRED_SYNCHRONIZED_SECONDARIES_TO_COMMIT:

```

--Run this script into Primary AOAG
ALTER AVAILABILITY GROUP [distributed-aoag]
SET (REQUIRED_SYNCHRONIZED_SECONDARIES_TO_COMMIT = 1);

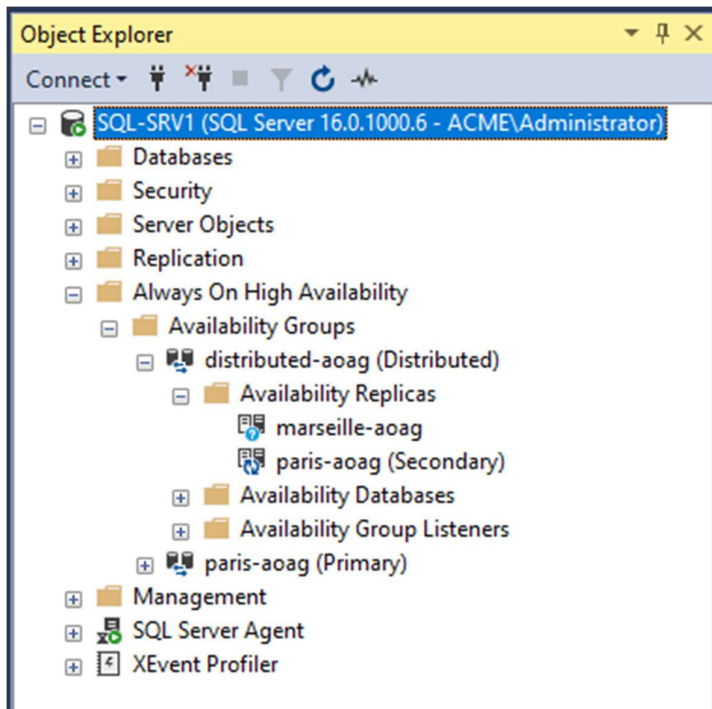
```

And now you are ready to change the role of the Primary AOAG (primary to secondary):

```

--Run this script into Primary AOAG, this will terminate client applications connected to the
primary replica of the primary AOAG
USE MASTER;
ALTER AVAILABILITY GROUP [distributed-aoag] SET (ROLE = SECONDARY);

```



1.6.5 Step 4

Change the role of the Secondary AOAG (Secondary to Primary), this command will change the role of the Secondary AOAG (secondary to primary) opening read/write the database, and also the AOAG roles into the Distributed AOAG is changed:

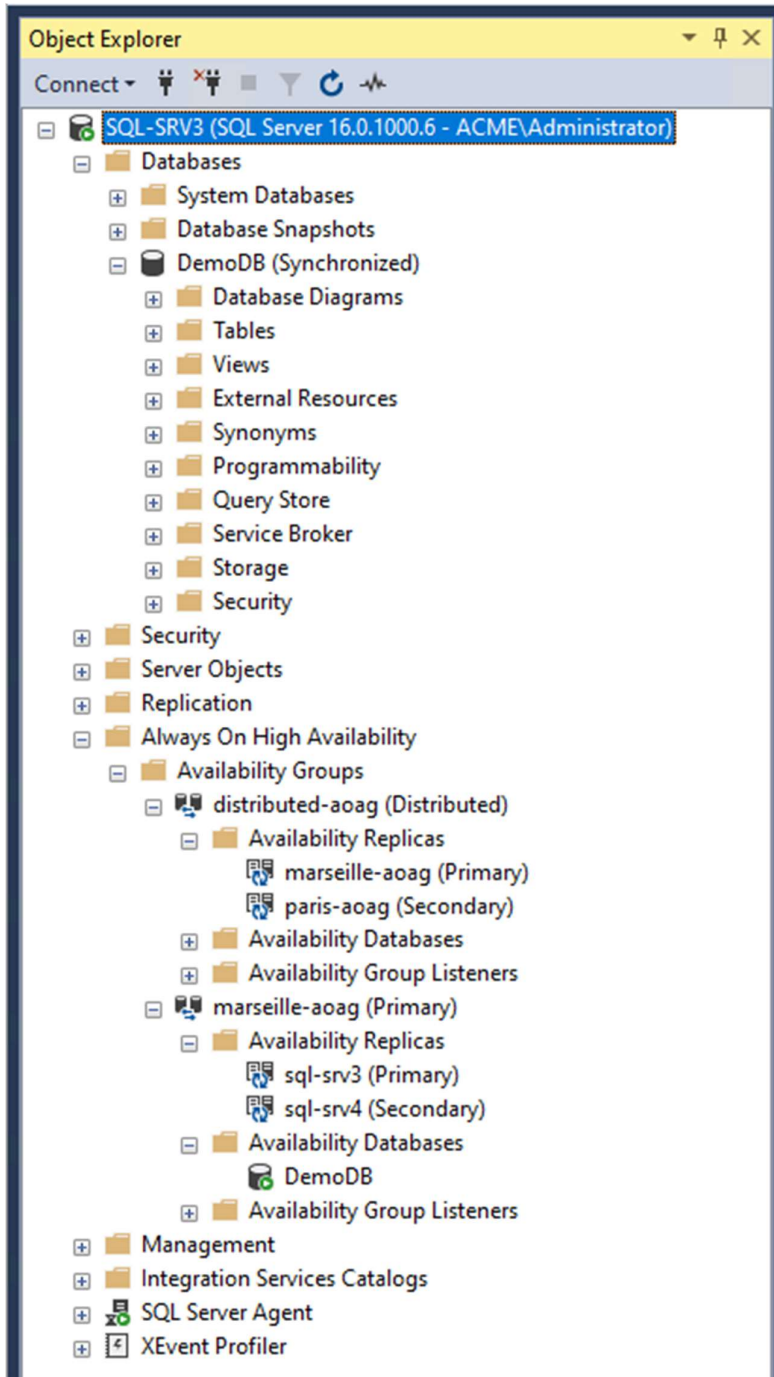
--Run this script into Secondary AOAG, this will terminate client applications connected to the primary replica of the primary AOAG

```
ALTER AVAILABILITY GROUP [distributed-aoag] FORCE_FAILOVER_ALLOW_DATA_LOSS;
```

And check the status:

```
--check the status on the new primary (formerly standby site)
select ag.name, ag.is_distributed, ar.replica_server_name, ar.availability_mode_desc,
ars.connected_state_desc, ars.role_desc,
ars.operational_state_desc, ars.synchronization_health_desc from sys.availability_groups
ag
join sys.availability_replicas ar on ag.group_id=ar.group_id
left join sys.dm_hadr_availability_replica_states ars
on ars.replica_id=ar.replica_id
where ag.is_distributed=1
GO
```

	name	is_distributed	replica_server_name	availability_mode_desc	connected_state_desc	role_desc	operational_state_desc	synchronization_health_desc
1	distributed-aoag	1	paris-aoag	SYNCHRONOUS_COMMIT	CONNECTED	SECONDARY	NULL	HEALTHY
2	distributed-aoag	1	marseille-aoag	SYNCHRONOUS_COMMIT	CONNECTED	PRIMARY	ONLINE	HEALTHY



1.6.6 Step 5

Now on the new Primary *UNSET* `REQUIRED_SYNCHRONIZED_SECONDARIES_TO_COMMIT`

```
--Run this script into Primary AOAG
ALTER AVAILABILITY GROUP [distributed-aoag]
SET (REQUIRED_SYNCHRONIZED_SECONDARIES_TO_COMMIT = 0);
```

And change the Availability Mode to return to the standard mode, execute this script on the Primary and on Secondary site:

```
--Run this first on the primary replica of the primary AOAG and then again on the secondary AOAG
ALTER AVAILABILITY GROUP [distributed-aoag]
MODIFY
AVAILABILITY GROUP ON
'paris-aoag' WITH
(
    AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT
),
'marseille-aoag' WITH
(
    AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT
);
```

1.7 Fallback procedure

To have Paris as Primary site and Marseille as Secondary site as at the beginning, simply execute a new switchover to invert the synchronization as described into failover procedure topic.