# Procedimento para lidar com funções e prioridades de gerentes de sessão em um conjunto de réplicas CPS

## Contents



## Introduction

Este documento descreve o procedimento para mover o sessionmgr da função principal e alterar a prioridade do sessionmgr em um Conjunto de Réplicas do Cisco Policy Suite (CPS).

## Prerequisites

### Requirements

A Cisco recomenda que você tenha conhecimento destes tópicos:

- Linux
- CPS
- MongoDB

A Cisco recomenda que você tenha acesso de raiz privilegiado à CLI do CPS.

### Componentes Utilizados

As informações neste documento são baseadas nestas versões de software e hardware:

- CPS 20.2
- MongoDB v3.6.17
- UCS-B

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Se a rede estiver ativa, certifique-se de que você entenda o impacto potencial de qualquer comando.

# Informações de Apoio

O CPS usa o MongoDB, onde mongood processes é executado na máquina virtual (VMs) do Sessionmgr para constituir sua estrutura básica de banco de dados. Possui vários conjuntos de réplicas para várias finalidades, que são ADMIN, Subscriber Profile Repository (SPR), BALANCE, SESSION, REPORTING e AUDIT.

Uma réplica definida no MongoDB é um grupo de processos mondeus que mantêm o mesmo conjunto de dados. Os conjuntos de réplica fornecem redundância e alta disponibilidade.Com várias cópias de dados em diferentes servidores de banco de dados, ele permite operações de leitura de compartilhamento de carga.

Um conjunto de réplicas contém vários nós que exibem dados e, opcionalmente, um nó de árbitro. Dos nós que suportam dados , um e apenas um membro é considerado o nó primário, enquanto os outros nós são considerados como nós secundários (um conjunto de réplicas pode ter vários secundários). O nó primário lida com todas as operações de gravação.

Os secundários replicam os logs de operação (log) do primário e aplicam as operações aos seus conjuntos de dados de modo que os conjuntos de dados dos secundários reflitam o conjunto de dados do primário. Se o principal não estiver disponível, um secundário qualificado terá uma eleição para eleger o novo primário. Um árbitro participa das eleições, mas não armazena dados.

Para obter o status dos conjuntos de réplica, execute o comando **diagnostics.sh —get_r** do ClusterManager ou do pcrfclient.

Fornecido aqui é um exemplo de conjunto de réplicas. **set07**.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-------------------------------------------------------------------------------------------------
---------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-------------------------------------------------------------------------------------------------
---------------------------------------|
```

Para obter informações de configuração do conjunto de réplicas, use estas etapas.

Etapa 1. Faça login no membro principal do MongoDB desse Conjunto de Réplicas. Execute este comando a partir do ClusterManager.

```
Command template:
#mongo --host <sessionmgrXX> --port <Replica Set port>

Sample command:
#mongo --host sessionmgr02 --port 27727
```

Etapa 2. Execute o comando para obter as informações de configuração do conjunto de réplicas.

```
set07:PRIMARY> rs.conf()
{
```

```
"_id" : "set07",
"version" : 2,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 2,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 3,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:PRIMARY>
```

**Note**: O Sessionmgr com a prioridade mais alta em um Conjunto de Réplicas funciona como membro Principal.

# Problema

Suponha que um gerente de sessão execute a função de um membro primário em um ou mais Conjuntos de Réplicas e, nesses casos, você deve mover a função principal do Conjunto de Réplicas para outro gerente de sessão,

1. Sempre que você executar qualquer atividade que envolva esse desligamento de VM do sessionmgr, para uma transição suave.

2. Se a integridade do sessionmgr for degradada devido a algum motivo, para manter a função adequada do Conjunto de Réplicas com algum outro sessionmgr saudável.

# Procedimento para Mover o sessionmgr da Prioridade Principal e Alterar Sesssionmgr num Conjunto de Réplicas

## Abordagem 1

Aqui a prioridade do sessionmgr em um Conjunto de Réplicas foi alterada diretamente no nível MongoDB. Aqui estão os passos para retirar a sessão mg02 de uma função primária no **set07**.

Opção 1. Altere a prioridade do sessionmgr02.

Etapa 1. Faça login no membro principal do MongoDB desse Conjunto de Réplicas.

```
Command template:
#mongo --host <sessionmgrXX> --port <Replica Set port>

Sample command:
#mongo --host sessionmgr02 --port 27727
```

Etapa 2. Execute o comando para obter as informações de configuração do conjunto de réplicas.

```
set07:PRIMARY> rs.conf()
{
"_id" : "set07",
"version" : 2,
"members" : [
{
"_id" : 0, -----> Position 0
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 2,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1, -----> Position 1
```

```
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2, -----> Position 2
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 3,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:PRIMARY>
```

**Note**: Anote a posição do respectivo gerente de sessão na saída rs.conf().

Etapa 3. Execute esse comando para mover o terminal para o modo de configuração.

```
set07:PRIMARY> cfg = rs.conf()
{
"_id" : "set07",
"version" : 2,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 2,
"tags" : {
```

```
},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 3,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:PRIMARY>
```

Etapa 4. Execute este comando para alterar a prioridade do sessionmgr.

```
Command template:
cfg.members[X].priority = X --> put the position here in [].

sample command:
cfg.members[2].priority = 1
```

Aqui o sessionmgr02 é atualmente o membro principal e sua posição é **2** e a prioridade é **3**.

Para mover este sessionmgr02 para fora da função primária, forneça o número de prioridade mais baixo maior que 0, mas menor que a prioridade de um membro secundário que tenha a prioridade

mais alta, por exemplo. **1** neste comando.

```
set07:PRIMARY> cfg.members[2].priority = 1
1
set07:PRIMARY>
```

Etapa 5. Execute este comando para confirmar a alteração.

```
set07:PRIMARY> rs.reconfig(cfg)
{
"ok" : 1,
"operationTime" : Timestamp(1641528658, 1),
"$clusterTime" : {
"clusterTime" : Timestamp(1641528658, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
}
2022-01-07T04:10:57.280+0000 I NETWORK [thread1] trying reconnect to sessionmgr02:27727
(192.168.10.140) failed
2022-01-07T04:10:57.281+0000 I NETWORK [thread1] reconnect sessionmgr02:27727 (192.168.10.140)
ok
set07:SECONDARY>
```

Etapa 6. Execute o comando novamente para verificar as alterações na prioridade do sessionmgr.

```
set07:SECONDARY> rs.conf()
{
"_id" : "set07",
"version" : 3,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 2,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
```

```
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 1, --> Here priority has been changed from 3 to 1.
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:SECONDARY>
```

Passo 7. Execute o comando **diagnostics.sh —get_r** do ClusterManager ou pcrfclient para verificar as alterações no status do Conjunto de réplicas.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC - PRIORITY |
-------------------------------------------------------------------------------------------
----------------------------------------
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - PRIMARY - sessionmgr01 - ON-LINE - -------- - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - SECONDARY - sessionmgr02 - ON-LINE - 0 sec - 1 |
|-------------------------------------------------------------------------------------------
----------------------------------------|
```

Agora, você vê que o sessionmgr02 foi transferido para secundário. Para tornar o sessionmgr02 um membro principal novamente, execute as Etapas 1 acima. 5. com esse comando na Etapa 4.

cfg.member[2].priority = Qualquer número maior que 2 mas menor que 1001 —> coloque a prioridade mais alta que a do membro principal atual, que é **2** na amostra.

```
set07:PRIMARY> cfg.members[2].priority = 5
5
set07:PRIMARY> rs.reconfig(cfg)
{
"ok" : 1,
"operationTime" : Timestamp(1641531450, 1),
```

```
"$clusterTime" : {
"clusterTime" : Timestamp(1641531450, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
}
2022-01-07T04:57:31.247+0000 I NETWORK [thread1] trying reconnect to sessionmgr01:27727
(192.168.10.139) failed
2022-01-07T04:57:31.247+0000 I NETWORK [thread1] reconnect sessionmgr01:27727 (192.168.10.139)
ok
set07:SECONDARY>
```

Execute o comando para verificar as alterações na prioridade do sessionmgr.

```
set07:SECONDARY> rs.conf()
{
"_id" : "set07",
"version" : 4,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 2,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 5, --> Here priority has been changed from 1 to 5.
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
```

```
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:SECONDARY>
```

Execute o comando **diagnostics.sh —get_r** do ClusterManager ou pcrfclient para verificar as alterações no status do Conjunto de réplicas.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-------------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 14 sec - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 5 |
|-------------------------------------------------------------------------------------------------
----------------------------------------|
```

Agora, você pode ver que o sessionmgr02 se tornou primário novamente.

Opção 2. Altere a prioridade de outro gerente de sessão secundário para torná-lo um membro primário. Aqui está a sessão mgr01.

Para tornar o sessionmgr01 membro principal, execute as etapas acima 1. 5. na Opção 1. com esse comando na Etapa 4.

cfg.member[0].priority = Qualquer número maior que 3 mas menor que 1001 —> Colocar prioridade maior que a do membro primário atual, que é "3" na amostra.

```
set07:PRIMARY> cfg.members[0].priority = 4
4
set07:PRIMARY> rs.reconfig(cfg)
{
"ok" : 1,
"operationTime" : Timestamp(1641540587, 1),
"$clusterTime" : {
"clusterTime" : Timestamp(1641540587, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
}
2022-01-07T07:29:46.141+0000 I NETWORK [thread1] trying reconnect to sessionmgr02:27727
(192.168.10.140) failed
2022-01-07T07:29:46.142+0000 I NETWORK [thread1] reconnect sessionmgr02:27727 (192.168.10.140)
ok
```

```
set07:SECONDARY>
```

Execute o comando para confirmar as alterações.

```
set07:SECONDARY> rs.conf()
{
"_id" : "set07",
"version" : 4,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 4, --> Here priority has been changed from 2 to 4.
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 3,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
```

```
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:SECONDARY>
```

Execute o comando **diagnostics.sh —get_r** do Cluster Manager ou do pcrfcleint para verificar as alterações no status do Conjunto de réplicas.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-------------------------------------------------------------------------------------------------------
---------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - PRIMARY - sessionmgr01 - ON-LINE - -------- - 4 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - SECONDARY - sessionmgr02 - ON-LINE - 0 sec - 3 |
|-------------------------------------------------------------------------------------------------------
---------------------------------------|
```

Agora, você pode ver que o sessionmgr01 se tornou primário, enquanto o sessionmgr02 se tornou secundário.

Para tornar o sessionmgr02 um membro principal novamente, execute as Etapas 1 acima. 5. na **Opção 1** com este comando na Etapa 4.

cfg.member[0].priority = Qualquer número menor que 3 mas maior que 0 —> Colocar prioridade menor que a do sessionmgr02 que é "3" na amostra.

```
set07:PRIMARY> cfg.members[0].priority = 1
1
set07:PRIMARY> rs.reconfig(cfg)
{
"ok" : 1,
"operationTime" : Timestamp(1641531450, 1),
"$clusterTime" : {
"clusterTime" : Timestamp(1641531450, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
}
2022-01-07T08:34:31.165+0000 I NETWORK [thread1] trying reconnect to sessionmgr01:27727
(192.168.10.139) failed
2022-01-07T08:34:31.165+0000 I NETWORK [thread1] reconnect sessionmgr01:27727 (192.168.10.139)
ok
set07:SECONDARY>
```

Execute esse comando para verificar as alterações na prioridade do sessionmgr.

```
set07:SECONDARY> rs.conf()
{
"_id" : "set07",
"version" : 4,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
```

```
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 1, --> Here priority has been changed from 4 to 1.
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 3,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:SECONDARY>
```

Execute o comando **diagnostics.sh —get_r** do ClusterManager ou pcrfclient para verificar as alterações no status do Conjunto de réplicas.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-------------------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
```

```
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 14 sec - 1 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-----------------------------------------------------------------------------------------
---------------------------------------|
```

Agora, você pode ver que o sessionmgr02 se tornou primário, enquanto o sessionmgr01 é secundário.

## Abordagem 2

Você pode usar o script CPS **set_priority.sh** do ClusterManager para alterar a prioridade do sessionmgr em um Conjunto de Réplicas. Por padrão, a prioridade dos membros é definida em ordem (com prioridade mais alta) como definido em **/etc/broadhop/mongoConfig.cfg** no ClusterManager.

Tome set07, por exemplo.

```
[root@installer broadhop]# cat mongoConfig.cfg
[SESSION-SET2]
SETNAME=set07
OPLOG_SIZE=5120
ARBITER=arbitervip:27727
ARBITER_DATA_PATH=/var/data/sessions.7
MEMBER1=sessionmgr02:27727
MEMBER2=sessionmgr01:27727
DATA_PATH=/var/data/sessions.1/2
[SESSION-SET2-END]
```

Para obter o status do conjunto de réplicas, execute o comando **diagnostics.sh —get_r** do ClusterManager ou do pcrfclient.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC - PRIORITY |
|-----------------------------------------------------------------------------------------
---------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-----------------------------------------------------------------------------------------
---------------------------------------|
```

Quando você compara os resultados mencionados acima, você pode ver que o sessionmgr02 é o 1º membro[Member1] do set07 em **/etc/broadhop/mongoConfig.cfg** , portanto o sessionmgr02 é o membro principal do set07 por padrão.

Fornecido aqui estão as opções de Alta Disponibilidade do CPS que usam o script **set_priority.sh** para mover o sessionmgr02 para fora da função de membro principal em set07.

Etapa 1. Defina a prioridade em ordem crescente.

```
Command template:
sh set_priority.sh --db arg --replSet arg --asc
```

```
where ,
--db arg --> arg is database name
[all|session|spr|admin|balance|report|portal|audit|bindings|session_configs|bindings_configs|spr
_configs]

--replSet arg -->arg is <setname>

Sample command:
sh set_priority.sh --db session --replSet set07 --asc

[root@installer ~]# sh set_priority.sh --db session --replSet set07 --asc

Set priorities is in progress. check log /var/log/broadhop/scripts/set_priority.log to know the
status
Setting priority for Replica-Set: SESSION-SET2
INFO Parsing Mongo Config file
INFO Priority set operation is completed for SESSION-SET2
INFO Priority set to the Database members is finished
INFO Validating if Priority is set correctly for Replica-Set: SESSION-SET2
WARNING Mongo Server trying to reconnect while getting config. Attempt #1
INFO Validated Priority is set correctly for Replica-Set: SESSION-SET2
Primary member sessionmgr01:27727 found for Replica SESSION-SET2

Set priorities process successfully completed.

[root@installer ~]#
```

Etapa 2. Execute o comando **diagnostics.sh —get_r** do ClusterManager ou do pcrfclient para verificar a alteração.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC - PRIORITY |
|-------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - PRIMARY - sessionmgr01 - ON-LINE - -------- - 3 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - SECONDARY - sessionmgr02 - ON-LINE - 0 sec - 2 |
|-------------------------------------------------------------------------------------------
----------------------------------------|
```

Agora, o sessionmgr01 tornou-se o membro principal, pois uma prioridade foi definida na ordem crescente, conforme definido em **/etc/broadhop/mongoConfig.cfg**.

Para tornar o sessionmgr02 um membro principal novamente, execute este comando.

```
[root@installer ~]# sh set_priority.sh --db session --replSet set07

Set priorities is in progress. check log /var/log/broadhop/scripts/set_priority.log to know the
status
Setting priority for Replica-Set: SESSION-SET2
INFO Parsing Mongo Config file
INFO Priority set operation is completed for SESSION-SET2
INFO Priority set to the Database members is finished
INFO Validating if Priority is set correctly for Replica-Set: SESSION-SET2
WARNING Mongo Server trying to reconnect while getting config. Attempt #1
INFO Validated Priority is set correctly for Replica-Set: SESSION-SET2
Primary member sessionmgr02:27727 found for Replica SESSION-SET2

Set priorities process successfully completed.
```

```
[root@installer ~]#
```

**Note**: Por padrão, a prioridade foi definida em ordem decrescente.

Execute o comando **diagnostics.sh —get_r** do ClusterManager ou do pcrfclient para verificar a alteração.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC - PRIORITY |
|-------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-------------------------------------------------------------------------------------------
----------------------------------------|
```

Agora, você pode ver que o sessionmgr02 se tornou primário, enquanto o sessionmgr01 é secundário.