Enhanced IP Allocation in Cisco Prime Access Registrar

This chapter describes the enhanced IP allocation feature in Cisco Prime Access Registrar (Prime Access Registrar).

In the previous versions of Prime Access Registrar, IP allocation happens internally based on a specific range of IPs configured. If there are multiple Prime Access Registrars in a deployment, each Prime Access Registrar server will have different range of IPs configured and can allocate/de-allocate IPs only within that specific range. Prime Access Registrar cannot allocate IPs from a common pool. This is addressed by the enhanced IP allocation feature.

With this feature, IP ranges will be read from the configuration and the common IP pools will be maintained in a centralized Mongo Database (MongoDB). Any Prime Access Registrar server which is connected to the DB can allocate an available IP for a user from the common IP pools. When the user disconnects, the IP is released back to the pool again. Along with the IP pools, the user sessions will also be maintained in centralized MongoDB.

The MongoDB version used for this feature is 3.6.2.

With the enhanced IP allocation feature, IPV6 address allocation is also supported.

Note This feature is supported only in CLI.

This chapter contains the following sections:

MongoDB Support

This section describes the MongoDB server features:

- The centralized DB can be a single MongoDB server, a MongoDB replica set, or a MongoDB shard.

- Replica set has one primary server and two or more secondary servers. The secondary servers acts as backup servers. Prime Access Registrar supports MongoDB cluster setup, that contains multiple shards (multiple replica sets).
- MongoDB has automatic failover mechanism. If the primary goes down, the election process is triggered among the available secondary servers. The new primary is elected and it starts processing the traffic.
- The secondary DB servers can be placed in the primary DB site as well as in the geographically distant failover sites for local DB failover and site failover.

IP Allocation Methodology

With the enhanced IP allocation feature, Prime Access Registrar provides the following support:

- Dynamic allocation of IPv4 and IPv6 addresses from the common IP pool information kept in Mongo DB.
- Multiple IP pools, each with a maximum size of 16 million IPs, can be configured in Mongo DB.
- Prime Access Registrar allocates IPs from the IP pool in a fail-over manner for the incoming RADIUS and Diameter requests.
- It is possible to select and allocate IPs from one particular IP pool using the scripting point.
- Multiple Prime Access Registrar servers can be connected to the Mongo DB for the IP allocation based on the requirement.
- IP allocation/de-allocation requests can be load-balanced to any Prime Access Registrar.
- Prime Access Registrar uses compressed format to store and retrieve the IPs from DB for effective use of DB resources.
- Prime Access Registrar supports MongoDB cluster deployments to meet higher scalability needs.
- MongoDB replica set provides fail-over capabilities with the primary and secondary nodes.
- Framed-IP-Address attribute holds the allocated IP address in the Access-Accept message.

Configuration Details

In Prime Access Registrar, a new type of session manager is introduced to support this feature. This session manager can handle both RADIUS/Diameter requests coming from the RADIUS/Diameter clients respectively. All the Prime Access Registrar servers connected to the same MongoDB/MongoDB replica set/MongoDB cluster must have the same session manager configuration.

Sample IPv6 Configuration:

```
--> cd /r
[ //localhost/Radius ]
    Name = Radius
    Description =
    Version =
    IncomingScript~ =
    OutgoingScript~ =
    DefaultAuthenticationService~ = null
    DefaultAuthorizationService~ = null
```

```
    DefaultAccountingService~ = local-file
    DefaultSessionService~ =
    DefaultSessionManager~ = MongoSessionManager
    UserLists/
    IPAddressAllocators/


--> cd IPAddressAllocators/
[ //localhost/Radius/IPAddressAllocators ]
Entries 1 to 1 from 1 total entries
Current filter: <all>
allocator1/

--> cd allocator1/
[ //localhost/Radius/IPAddressAllocators/allocator1 ]
    Name = allocator1
    Description =
    Type = mongo
    IPAllocationType = IPv6
    DepletedPoolTimeOut = 2M
    IPRecordTimeOut = 1M

    IPAddressPools/

--> cd IPAddressPools/
[ //localhost/Radius/IPAddressAllocators/allocator1/IPAddressPools ]
Entries 1 to 1 from 1 total entries
Current filter: <all>

P1/

--> cd P1
[ //localhost/Radius/IPAddressAllocators/allocator1/IPAddressPools/P1 ]
    Name = P1
    Description =
    Identifier = 0
    Type = ipv6
    StartIPv6 = 2025::20c:29ff:fe65:9802
    EndIPv6 = 2025::20c:29ff:feff:ffff
    IPv6Prefix = 2025::/64

--> cd /r/SessionManagers/MongoSessionManager/
[ //localhost/Radius/SessionManagers/MongoSessionManager ]
    Name = MongoSessionManager
    Description =
    Type = geo
    EnableDiameter = FALSE
    IncomingScript =
    OutgoingScript =
    AllowAccountingStartToCreateSession = FALSE
    SessionTimeOut =
    PhantomSessionTimeOut =
    SessionKey = User-Name
    ResourceManagers/

--> cd ResourceManagers/
[ //localhost/Radius/SessionManagers/MongoSessionManager/ResourceManagers ]
    1. ipv6

--> cd /r/ResourceManagers/
[ //localhost/Radius/ResourceManagers ]
    Entries 1 to 6 from 6 total entries
    Current filter: <all>
```

```
    IPA-Pool/
    IPA-Pool-2/
    ipv6/
    IPX-Pool/
    Per-Group/
    Per-User/

--> cd Ipv6
[ //localhost/Radius/ResourceManagers/ipv6 ]
    Name = ipv6
    Description =
    Type = geo-ipv6-dynamic
    IPv6Prefix = 2025::/64
    ReuseIPForSameSessionKeyAndUser = TRUE
    IPAllocator = allocator1
```

Sample IPv4 Configuration:

```
--> cd allocator1/
[ //localhost/Radius/IPAddressAllocators/allocator1 ]
    Name = allocator1
    Description =
    Type = mongo
    DepletedPoolTimeOut = 2M
    IPRecordTimeOut = 1M
    IPAllocationType = IPv4
    IPAddressPools/

--> cd IPADDressPools/
[ //localhost/Radius/IPAddressAllocators/allocator1/IPAddressPools ]
    Entries 1 to 4 from 4 total entries
    Current filter: <all>
    P1/
    P2/
    P3/
    P4/

[ //localhost/Radius/IPAddressAllocators/allocator1/IPAddressPools ]
    Entries 1 to 4 from 4 total entries
    Current filter: <all>


P1/
    Name = P1
    Description =
    Identifier = 10
    Type = ipv4
    NetMask = 255.0.0.0
    Start = 10.0.0.0
    End = 10.255.255.255


P2/
    Name = P2
    Description =
    Identifier = 20
    Type = ipv4
    NetMask = 255.0.0.0
    Start = 11.0.0.0
    End = 11.255.255.255

P3/
    Name = P3
```

```
    Description =
    Identifier = 30
    Type = ipv4
    NetMask = 255.0.0.0
    Start = 12.0.0.0
    End = 12.255.255.255

P4/
    Name = P4
    Description =
    Identifier = 40
    Type = ipv4
    NetMask = 255.0.0.0
    Start = 13.0.0.0
    End = 13.255.255.255

[ //localhost/Radius/ResourceManagers ]
Entries 1 to 8 from 8 total entries
Current filter: <all>

    geo-per-user/
    IPA-Pool/
    IPA-Pool-2/
    ipv4/
    ipv6/
    IPX-Pool/
    Per-Group/
    Per-User/

--> cd ipv4
[ //localhost/Radius/ResourceManagers/ipv4 ]
    Name = ipv4
    Description =
    Type = geo-ipv4-dynamic
    NetMask = 255.0.0.0
    ReuseIPForSameSessionKeyAndUser = FALSE
     IPAllocator = allocator1

[ //localhost/Radius/SessionManagers/MongoSessionManager ]
    Name = MongoSessionManager
    Description =
    Type = geo
    EnableDiameter = TRUE
    IncomingScript =
    OutgoingScript =
    AllowAccountingStartToCreateSession = FALSE
    SessionTimeOut =
    PhantomSessionTimeOut =
    SessionKey = User-Name
    SessionCreationCmdList = 265
    SessionDeletionCmdList = 275
    SessionRestorationTimeOut =
    ResourceManagers/

--> cd ResourceManagers/
[ //localhost/Radius/SessionManagers/MongoSessionManager/ResourceManagers ]
    1. ipv4

--> cd /r/advanced/remotemongosessionServer/

[ //localhost/Radius/Advanced/RemoteMongoSessionServer ]
    ReactivateTimerInterval = 300000
    Timeout = 15
    MongoTimeOutCount = 10
```

```
    MongoActiveConnetionThresholdCount = 4
    MongoConnectionReactivationInterval = 3000
    DataSourceConnections = 4
    DataSource =
    SNMPTrapIP =
    SNMPTrapPort = 1521
    KeepAliveTimerInterval = 0

[ //localhost/Radius/Advanced/ODBCDataSources/mongo ]
    Name = mongo
    Description =
    Type = mongoc
    UserID =
    Password =
    DataBase =
    Server = 10.126.246.113:27017
    DBReadPreference = Nearest
    IsReplicaSet = FALSE
```

Table 11-1 lists the attributes added under /RADIUS/Advanced for the IP Allocation feature.

Table 11-1 /RADIUS/Advanced Attributes added for IP Allocation Feature

Property	Description
IPDataBackingStore SyncInterval	Interval at which the IP data is written to the backing store.
IPDataBackingStore PruneInterval	The sleep time interval of the IP data backing store pruning thread. The recommended and default value is six hours, but you can modify this based on the traffic patterns you experience. With IPDataBackingStorePruneInterval set to six hours, pruning will occur six hours after you restart or reload the Prime Access Registrar server and recur every six hours. You can set a very low value for this property to make pruning continuous, but there might not be enough data accumulated for the pruning to occur and pruning might be less effective compared to the default setting.
IPDataBackingStore DiscThreshold	Maximum size limit of any IP data log files generated; the default is 10 gigabytes. The value of IPDataBackingStoreDiscThreshold is made up of a number of units which can be K, kilobyte, or kilobytes, M, megabyte, or megabytes, or G, gigabyte, or gigabytes.
IPDataPurgeInterval	The interval in which Prime Access Registrar must check for timed-out IP records.
IPDocumentTimeOut	If there is any document in locked state for this timeout period, then those documents will be released/unlocked during the purge operation.

Server Monitoring for IP Allocation

Prime Access Registrar supports server monitoring for the IP allocation feature, using which high and low IP thresholds can be monitored. The following attributes are added to support this functionality:

- IPHighThreshold—Absolute integer value that indicates the maximum number of IPs that can be allocated by the server. Default is 0. When the number of IPs exceeds the given high threshold value, Prime Access Registrar generates a **carIPCapacityFull** trap.
- IPLowThreshold—Absolute integer value that indicates the minimum number of IPs that can be allocated by the server. Default is 0. After reaching the high threshold, if the number of IPs drops below a low threshold value, Prime Access Registrar generates a **carIPCapacityNotFull** trap.

For details about the **carIPCapacityFull** and **carIPCapacityNotFull** traps, refer to the "Using SNMP" chapter of the *Cisco Prime Access Registrar 9.0 User Guide*.

Common Configuration Setup

If there are multiple Prime Access Registrar servers in a deployment, common configuration must be maintained across all the servers. To maintain consistency with the configuration of all the Prime Access Registrar servers, a Python tool is developed and shipped with the Prime Access Registrar installation package. After installation, this Python tool (e.g. main.py) will be present in the **/cisco-ar/bin/** directory.

Note The Python tool will not work properly, if there is a CLI access from multiple terminals.

Note Also, ensure that the correct system time is maintained across all the Prime Access Registrar servers in a deployment.

After installing Prime Access Registrar in all the identified servers, follow the below steps to maintain common configuration across all Prime Access Registrar servers:

Step 1 Set the attribute **IsMaster** under **/r/advanced** in aregcmd to TRUE.

Step 2 Perform the IP allocation configuration through aregcmd CLI interface in any one of the Prime Access Registrar servers.

Step 3 Execute **SAVE** from aregcmd. This creates an XML file.

Following is a sample XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<ipaddressallocation>

   <ipsessionmanager>

      <sessionmanager ismodified = "-1" name = "geoSes" type = "geo" enablediameter =
"FALSE" incomingscript = "" outgoingscript = "" allowaccountingstarttocreatesession =
"FALSE" sessiontimeout = "" phantomsessiontimeout = "" sessionkey = ""
sessioncreationcommandlist = "" sessiondeletioncommandlist = ""/>
      <resourcemanagers>
         <rm name = "geo-per-user" index = "1"/>
      </resourcemanagers>

   </ipsessionmanager>


   <ipresourcemanager>
```

```
      <resourcemanager ismodified = "-1" name = "geo-ipv4" type = "geo-ipv4-dynamic"
netmask = "255.0.0.0" ipv6prefix = "" reuseipforsamesessionkeyanduser = "FALSE"
ipallocator = "A1"/>

      <resourcemanager ismodified = "-1" name = "geo-per-user" type =
"geo-user-session-limit" usersessionlimit = "1"/>

   </ipresourcemanager>

   <ipallocators>

      <allocator ismodified = "-1" name = "A1" type = "mongo" ipallocationtype = "IPv4"/>

      <ipallocationpool allocator = "A1" name = "P1" identifier = "10" type = "ipv4"
netmask = "255.0.0.0" start = "10.0.0.0" end = "10.255.255.255"/>
      <ipallocationpool allocator = "A1" name = "P2" identifier = "20" type = "ipv4"
netmask = "255.0.0.0" start = "11.0.0.0" end = "11.255.255.255"/>
      <ipallocationpool allocator = "A1" name = "P3" identifier = "30" type = "ipv4"
netmask = "255.0.0.0" start = "12.0.0.0" end = "12.255.255.255"/>
      <ipallocationpool allocator = "A1" name = "P4" identifier = "40" type = "ipv4"
netmask = "255.0.0.0" start = "13.0.0.0" end = "13.255.255.255"/>

   </ipallocators>
</ipaddressallocation>
```

Step 4 Run the Python tool:

```
- python /cisco-ar/bin/main.py
```

The tool will do the following:

- Prompt for the total number of Prime Access Registrar servers connected to the DB. Enter hte number.
- Convert the generated XML into a **.rc** file.

Following is a sample .rc file.

```
delete /Radius/SessionManagers/MongoSession
add /Radius/SessionManagers/MongoSession
set /Radius/SessionManagers/MongoSession/type geo
set /Radius/SessionManagers/MongoSession/enablediameter FALSE
set /Radius/SessionManagers/MongoSession/incomingscript skip
set /Radius/SessionManagers/MongoSession/allowaccountingstarttocreatesession FALSE
set /Radius/SessionManagers/MongoSession/sessionkey User-Name:Nas-Port
add /Radius/SessionManagers/geoSes
set /Radius/SessionManagers/geoSes/type geo
set /Radius/SessionManagers/geoSes/enablediameter FALSE
set /Radius/SessionManagers/geoSes/allowaccountingstarttocreatesession FALSE
set /Radius/SessionManagers/MongoSession/ResourceManagers/1 geo-ipv4
set /Radius/SessionManagers/geoSes/ResourceManagers/1 geo-per-user
add /Radius/ResourceManagers/geo-ipv4 "" geo-ipv4-dynamic
set /Radius/ResourceManagers/geo-ipv4/netmask 255.0.0.0
set /Radius/ResourceManagers/geo-ipv4/reuseipforsamesessionkeyanduser FALSE
set /Radius/ResourceManagers/geo-ipv4/ipallocator A1
add /Radius/ResourceManagers/geo-per-user "" geo-user-session-limit 1
save
```

Step 5 Restart Prime Access Registrar. This will initialize and create the following:

- Collections in the MongoDB—These are the names of the configured session managers. These collections are created inside the DB, which is configured in mongoc data source configuration in aregcmd.

Note Make sure you do not delete the database name and collections to avoid possible data inconsistency issue.

- Required indexes in all the collections for faster access
- The DB named **IPProvisioning**.

  Note Both the IPProvisioning database and the database configured under mongoc data source in aregcmd must have the same credentials.

- Pools in the **IPProvisioning** DB based on the **IPAddressAllocators** configuration

Step 6 Once initialization is done, the Python tool resets the **IsMaster** attribute to FALSE in aregcmd and prompts for the IP, credentials, etc., of the next Prime Access Registrar server. Provide the required details in the tool.

Step 7 After getting the credentials, the Python tool logs in to the new Prime Access Registrar server and dumps the **.rc** file generated. It also prompts you to restart the Prime Access Registrar server.

Step 8 Enter **Yes** and restart the Prime Access Registrar server.

Step 9 Repeat the above three steps for all the Prime Access Registrar servers. This way, configuration is maintained consistently across all individual Prime Access Registrar servers.

Sample IP Allocation Traces

Following are the sample IPv4 allocation and de-allocation traces:

Enhanced IP Allocation – Sample IPv4 Allocation Traces

```
01/15/2019 18:47:10.572: P78: SessionManager MongoSessionManager created Session S2
01/15/2019 18:47:10.572: P78: Session S2, Session-Start-Time: 01/15/2019 18:47:10, NAS:
localhost, NAS-Port: 1, User-Name: bob, Session-Key: bob
01/15/2019 18:47:10.572: P78: ResourceManager ipv4: Requesting allocator allocator2 to
allocate an ipv4 address
01/15/2019 18:47:10.572: P78: MongoIPAllocator allocator2: address not available in local
store P2
01/15/2019 18:47:10.572: P78: MongoIPAllocator allocator2: sending request to the
RemoteMongoServer Internal-Mongo-Server
01/15/2019 18:47:10.573: P78: MonogIPAllocator allocator2: Database returned Bitmap:0
Index:0
01/15/2019 18:47:10.573: P78: MonogIPAllocator allocator2: Successfully stored the bitmap
in the localstore P2
01/15/2019 18:47:10.573: P78: MonogIPAllocator allocator2: Allocating IP from Bitmap:0
Index:0
01/15/2019 18:47:10.593: P78: MongoIPAllocator allocator2: Successfully allocated an ip
address from pool P2
01/15/2019 18:47:10.593: P78: MongoIPAllocator allocator2:Allocation completed and Need to
update to database01/15/2019 18:47:10.594: P78: MongoIPAllocator allocator2: Successfully
allocated IPAddress
01/15/2019 18:47:10.594: P78: IPResourcManager ipv4:Allocator returned success for ipv4
address allocation request
01/15/2019 18:47:10.594: P78: ResourceManager ipv4 allocated a resource to Session S2:
Allocated IP Address 10.0.0.0
```

```
01/15/2019 18:47:10.594: P78: Writing Session S2(bob) to the mongo database.
01/15/2019 18:47:10.594: P78: Session Count Update 0
01/15/2019 18:47:10.594: P78: The collection name is MongoSessionManager
01/15/2019 18:47:10.594: Log: Collection handle created : MongoSessionManager
01/15/2019 18:47:10.594: Remote Mongo Session Server (Connection 10):
MongoActiveConnectionCount = 32 and ConnectionTimedOutCount = 0
01/15/2019 18:47:10.595: Running AddSession Script:
01/15/2019 18:47:10.595: P78: Releasing acquired Session S2(bob)
01/15/2019 18:47:10.595: P78: SessionManager MongoSessionManager done with packet
01/15/2019 18:47:10.595: P78: Trace of Access-Accept packet
01/15/2019 18:47:10.595: P78:    identifier = 1
01/15/2019 18:47:10.595: P78:    length = 32
01/15/2019 18:47:10.595: P78:    respauth =
d3:5c:cc:73:7d:6b:17:fd:f1:0e:21:9d:90:bc:83:1f
01/15/2019 18:47:10.595: P78:    Framed-IP-Address = 10.0.0.0
01/15/2019 18:47:10.595: P78:    Framed-IP-Netmask = 255.0.0.0
01/15/2019 18:47:10.595: P78: Sending response to 127.0.0.1
01/15/2019 18:47:10.595: P78: Packet successfully removed
01/15/2019 18:47:10.595: P78: Packet Deleted
```

Enhanced IP Allocation – Sample IPv4 De-Allocation Traces

```
01/15/2019 18:49:09.741: R2: ResourceManager ipv4 allocated a resource to Session S2:
Resurrected session with IP Address 10.0.0.0
01/15/2019 18:49:09.741: P80: Acquiring session for bob..., the request is from
localhost:1
01/15/2019 18:49:09.741: P80: Session S2(bob) acquired...
01/15/2019 18:49:09.741: P80: SessionManager MongoSessionManager decremented the
Accounting Counter for Session S2(bob), now -1
01/15/2019 18:49:09.741: P80: SessionManager MongoSessionManager is deleting Session
S2(bob)
01/15/2019 18:49:09.741: P80: Releasing Geo Resources
01/15/2019 18:49:09.741: P80: Entered releaseGeoResource
01/15/2019 18:49:09.741: P80: MongoAllocator allocator2: Releasing ip address 10.0.0.0 in
the mongodb
01/15/2019 18:49:09.741: P80: MongoIPAllocator allocator2: sending request to the
RemoteMongoServer Internal-Mongo-Server
01/15/2019 18:49:09.741: Log: Collection handle created : P2
01/15/2019 18:49:09.742: Remote Mongo Session Server (Connection 28):
MongoActiveConnectionCount = 32 and ConnectionTimedOutCount = 0
01/15/2019 18:49:09.742: P80: ResourceManager ipv4 released a resource from Session S2:
Released IP address 10.0.0.0
01/15/2019 18:49:09.742: P80: The collection name is MongoSessionManager
01/15/2019 18:49:09.742: Log: Collection handle created : MongoSessionManager
01/15/2019 18:49:09.742: Remote Mongo Session Server (Connection 25):
MongoActiveConnectionCount = 32 and ConnectionTimedOutCount = 0
01/15/2019 18:49:09.742: P80: Trace of Accounting-Response packet
01/15/2019 18:49:09.742: P80:    identifier = 2
01/15/2019 18:49:09.742: P80:    length = 20
01/15/2019 18:49:09.742: P80:    respauth =
37:07:c1:12:8f:28:ec:3e:9f:a1:df:cd:f1:99:92:65
01/15/2019 18:49:09.742: P80: Sending response to 127.0.0.1
```