CPS Geographic Redundancy Guide, Release 18.1.0 (Restricted Release) (1)

First Published: 2018-03-16

Last Modified: 2019-03-15

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com go trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2018–2019 Cisco Systems, Inc. All rights reserved.

CONTENTS

PREFACE

Preface ix

About this Guide ix

Audience ix

Additional Support ix

Conventions (all documentation) x

Communications, Services, and Additional Information xi

PREFACE

RESTRICTED RELEASE xiii

CHAPTER 1

Overview 1

CPS Architecture Overview 1

Operations, Administration and Management (OAM) 1

Three-tier Processing Architecture 1

Persistence Layer 3

Geographic Redundancy 5

Overview 5

Concepts 5

Active/Standby 5

Failure 6

Failover 6

Failover Time 6

Heartbeat 6

Split Brain 6

Cross-site Referencing 6

Arbiter 6

Data Redundancy 6

Operations Log 7

CHAPTER 2 GR Reference Models 9

GR Reference Models 9

Without Session Replication 9

Active/Standby 10

Active/Active 11

With Session Replication 12

Active/Standby 12

Active/Active 13

Advantages and Disadvantages of GR Models 14

SPR/Balance Considerations 15

SPR Considerations 15

Balance Considerations 15

Data Synchronization 16

Data Synchronization in MongoDB 17

CPS GR Dimensions 17

Different Databases 17

Number of Sites 18

Arbiter Considerations 18

Database Partitioning - Shards 18

Session Shard Considerations 19

Network Diagrams 19

Management Network Interface 21

External and Routable Network Interface 21

Replication Network Interface 21

Internal Network 22

Summary 22

Network Requirements 22

CHAPTER 3 GR Installation - VMware 23

GR Installation Process 23

Overview 23

Prerequisites 24

Reference for CPS VM and Host Name Nomenclature 25
Arbiter Installation 27
On Third Site 27
On Primary Site 29
Standalone Arbiter Deployment On VMware 30
Configure Remote/Peer Site VM 32
Session Manager VM Information on Local Site 32
Policy Director (lb) VM Information on Local Site 33
Database Configuration 34
Balance Backup Database Configuration 37
Session Cache Hot Standby 40
Prerequisites 41
Configuration 41
Failover Detection 42
Limitation 43
Policy Builder Configuration 43
Access Policy Builder from Standby Site when Primary Site is Down 46
qns.conf Configuration Changes for Session Replication 47
Configurations to Handle Database Failover when Switching Traffic to Standby Site Due to Load Balancer Fail/Down 48

CHAPTER 4 **GR Installation - OpenStack 51**
GR Installation - OpenStack 51
Arbiter Installation on OpenStack 55
Configuration Parameters - GR System 58
policyServerConfig 59
dbMonitorForQns and dbMonitorForLb 60
clusterInfo 61
Example Requests and Response 61

CHAPTER 5 **Geographic Redundancy Configuration 65**
Database Migration Utilities 65
Split Script 66
Audit Script 68

Recovery Procedures 69
Site Recovery Procedures 69
Manual Recovery 69
Automatic Recovery 70
Individual Virtual Machines Recovery 72
Database Replica Members Recovery Procedures 72
Automatic Recovery 72
Manual Recovery 75
Additional Session Replication Set on GR Active/Active Site 84
Rollback Additional Session Replication Set 94
Network Latency Tuning Parameters 96
Remote SPR Lookup based on IMSI/MSISDN Prefix 96
Prerequisites 96
Configuration 97
Remote Balance Lookup based on IMSI/MSISDN Prefix 98
Prerequisites 98
Configuration 98
SPR Provisioning 100
SPR Location Identification based on End Point/Listen Port 100
Prerequisites 100
Configuration 100
API Router Configuration 101
Use Cases 101
HTTP Endpoint 102
Configuration 103
Configurations to Handle Traffic Switchover 113
When Policy Server (QNS) is Down 113
When Replicated (inter-site) Database is not Primary on a Site 114
When Virtual IP (VIP) is Down 115
Configuring Session Database Percentage Failure 115
Remote Databases Tuning Parameters 116
SPR Query from Standby Restricted to Local Site only (Geo Aware Query) 117
Balance Location Identification based on End Point/Listen Port 120
Prerequisites 120

Configuration 120
Balance Query Restricted to Local Site 121
Session Query Restricted to Local Site during Failover 123
Publishing Configuration Changes When Primary Site becomes Unusable 126
Graceful Cluster Shutdown 128
Active/Active Geo HA - Multi-Session Cache Port Support 129
Install Geo HA 129
Enable Geo HA 130
Configuration 130
GR Configuration with Session Replication Across Sites 134
Local Session Affinity - Capacity Planning 134
Limitation 136
Handling RAR Switching 137
Configure Cross-site Broadcast Messaging 137
Example 138
Configure Redundant Arbiter (arbitervip) between pcrfclient01 and pcrfclient02 139
Moving Arbiter from pcrfclient01 to Redundant Arbiter (arbitervip) 140

CHAPTER 6 **GR Failover Triggers and Scenarios** 143
Failover Triggers and Scenarios 143
Site Outage 143
Gx Link Failure 145
Failover Time Improvement 147
Rx Link Failure 147
Load Balancer VIP Outage 148
Arbiter Failure 149

APPENDIX A **OpenStack Sample Files - GR** 151
Sample Heat Environment File 151
Sample Heat Template File 153
Sample YAML Configuration File - site1 178
Sample YAML Configuration File - site2 186
Sample Mongo Configuration File - site1 193
Sample Mongo Configuration File - site2 195

Sample Mongo GR Configuration File 196
Sample GR Cluster Configuration File - site1 199
Sample GR Cluster Configuration File - site2 199
Sample Set Priority File - site1 199
Sample Set Priority File - site2 199
Sample Shard Configuration File - site1 200
Sample Shard Configuration File - site2 200
Sample Ring Configuration File 200
Sample Geo Site Lookup Configuration File - site1 200
Sample Geo Site Lookup Configuration File - site2 201
Sample Geo-tagging Configuration File - site1 201
Sample Geo-tagging Configuration File - site2 201
Sample Monitor Database Configuration File - site1 201
Sample Monitor Database Configuration File - site2 202

Preface

- About this Guide, on page ix
- Audience, on page ix
- Additional Support, on page ix
- Conventions (all documentation), on page x
- Communications, Services, and Additional Information, on page xi

About this Guide

Welcome to Cisco Policy Suite (CPS) Geographic Redundancy (GR) Guide.

This document describes the Geographic Redundancy architecture. This document is intended as a starting point for learning about GR and how it works and also contains details about GR specific installations and configurations.

Audience

This guide is best used by these readers:

- Network administrators
- Network engineers
- Network operators
- System administrators

This document assumes a general understanding of network architecture, configuration, and operations.

Additional Support

For further documentation and support:

- Contact your Cisco Systems, Inc. technical representative.
- Call the Cisco Systems, Inc. technical support number.

- Write to Cisco Systems, Inc. at support@cisco.com.
- Refer to support matrix at https://www.cisco.com/c/en/us/support/index.html and to other documents related to Cisco Policy Suite.

Conventions (all documentation)

This document uses the following conventions.

Conventions	Indication
bold font	Commands and keywords and user-entered text appear in **bold** font.
italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
[ ]	Elements in square brackets are optional.
{x | y | z }	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
courier font	Terminal sessions and information the system displays appear in courier font.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Note Means reader take note. Notes contain helpful suggestions or references to material not covered in the manual.

Caution Means reader be careful. In this situation, you might perform an action that could result in equipment damage or loss of data.

Warning IMPORTANT SAFETY INSTRUCTIONS.

Means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device.

SAVE THESE INSTRUCTIONS

Note Regulatory: Provided for additional information and to comply with regulatory and customer requirements.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

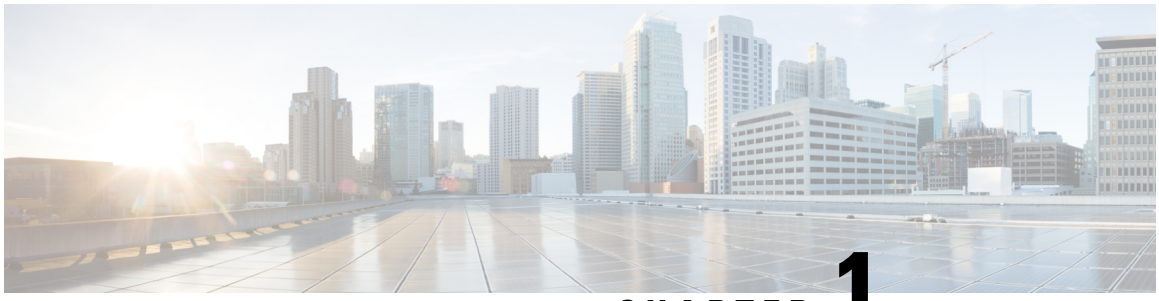
Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.

RESTRICTED RELEASE

Important This is a Short Term Support (STS) release with availability and use restrictions. Contact your Cisco Account or Support representatives for more information.

RESTRICTED RELEASE



CHAPTER 1

Overview

- CPS Architecture Overview, on page 1
- Geographic Redundancy, on page 5

CPS Architecture Overview

The Cisco Policy Suite (CPS) solution utilizes a three-tier virtual architecture for scalability, system resilience, and robustness consisting of an I/O Management, Application, and Persistence layers.

The main architectural layout of CPS is split into two main parts:

- Operations, Administration and Management (OAM)
- Three-tier Processing Architecture

Operations, Administration and Management (OAM)

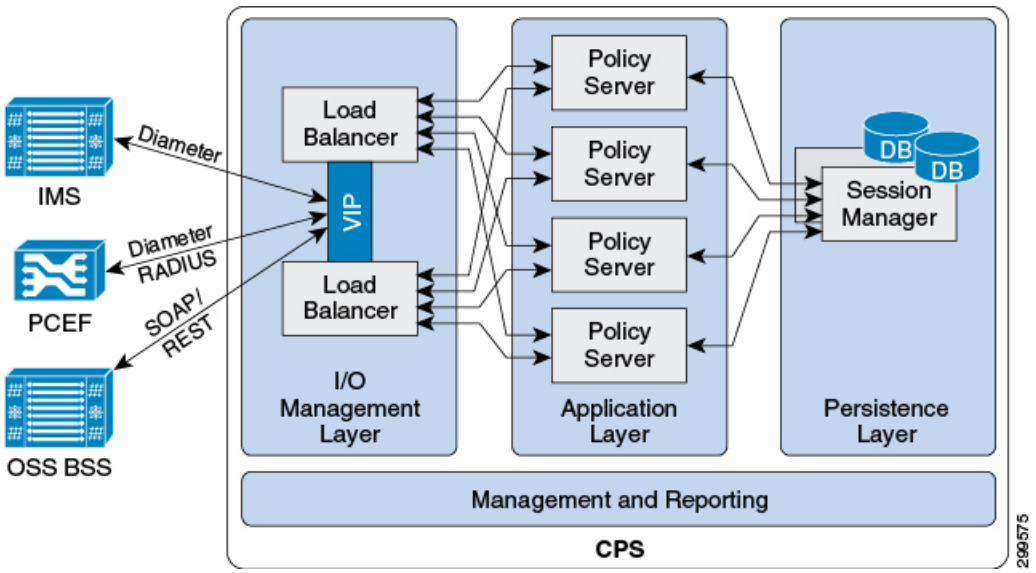
The OAM contains the CPS functions related to the initial configuration and administration of CPS.

- Operators use the Policy Builder GUI to define the initial network configuration and deploy customizations.
- Operators use the Control Center GUI or functionality that is provided through the Unified API to monitor day-to-day operations and manage the network. One of the primary management functions of Control Center is to manage subscribers. This aspect of Control Center is also referred to as the Unified Subscriber Manager (USuM).

Three-tier Processing Architecture

This section describes the three-tier architecture.

Figure 1: 3 Tier Architecture



The three-tier architecture defines how CPS handles network messages and gives CPS the ability to scale. The three processing tiers are:

- I/O Management Layer

  The I/O Management Layer handles I/O management and distribution within the platform. Load Balancer (LB) VMs, also referred to as Policy Director (PD) VMs implements I/O management layer functions. This layer supports internal load balancers to load balance requests to the relevant modules.

- Application Layer

  The Application Layer handles the transaction workload and does not maintain subscriber session state information. The main module of the Application Layer is a high performance rules engine.

- Persistence Layer

  The Persistence Layer consists of the Session Manager - a document-oriented database used to store session, subscriber information, and balance data (if and when applicable). Session Manager VMs implements this function.

  The databases that are included in Session Manager are:

  - Admin
  - Audit
  - Custom Reference Data
  - Policy Reporting
  - Sessions
  - Balance
  - SPR

For more information on Persistence Layer, refer to Persistence Layer, on page 3.

Persistence Layer

The Persistence Layer is responsible for storing session information, as well as subscriber profile and quota information if applicable. This is done using the Session Manager application. It is the persistence layer that maintains state within CPS. Geographic redundancy is achieved through data synchronization of the persistence layer between sites.

The Session Manager is built using MongoDB, a high-performance and high-availability document-oriented database.

The MongoDB obtains high performance by using a 'file-backed in-memory database'. To achieve high performance, the MongoDB stores as much of the data as possible in memory (and thus is very fast), but the data is mirrored and written out to disk to preserve the database information across restarts.

Access to the database is typically performed using the Unified API (SOAP/XML) interface. GUI access is typically limited to lab environments for testing and troubleshooting, and can be used to perform the following tasks:

- Manage subscriber data (if SPR used), that is, find, create or edit subscriber information
- Stop database or check the availability of subscriber sessions
- Review and manage subscriber sessions
- Populate custom reference data tables: Custom reference data tables allow service providers to create their own data structures that can be used during policy evaluation. Examples of information that can be included in custom reference data tables include:
  - Device parameters
  - Location data mapping (for example, map network sites and cell sites into the subscriber's home network)
  - Roaming network or preferred roaming network
  - IMEI data tagging for smart phone, Apple, android device, and so on

Unified Subscriber Manager (USuM)/Subscriber Profile Repository (SPR)

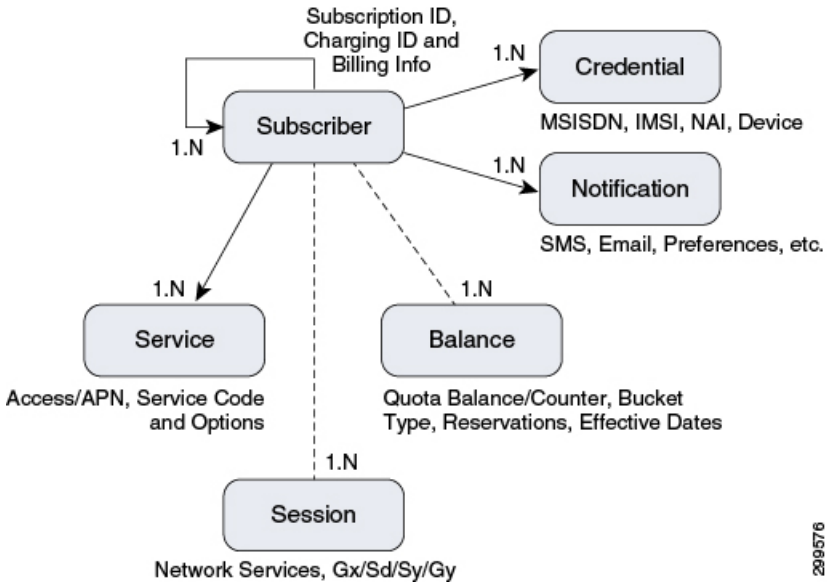
USuM manages subscriber data in a Subscriber Profile Repository (SPR). This includes the credentials with which a subscriber is able to log in and services allocated to the subscriber. The details of what a service means are stored in Policy Builder.

Each subscriber record that is stored in the USuM is a collection of the data that represents the real world end-subscriber to the system. Examples include which of the service provider's systems that the subscriber can access (mobile, broadband, and so on.) or to identify specific plans and service offerings that the subscriber can utilize.

Additionally, USuM can correlate balance and session data to a subscriber. Balance data is owned by Multi-Service Balance Manager (MsBM) and is correlated by the Charging Id. Session data is correlated by the credential on the session which should match an USuM credential. Session data is managed by CPS core and can be extended by components.

In 3GPP terminology, the USuM is a Subscriber Profile Repository (SPR). The following is a symbolic representation of how data portions of the Cisco SPR relate and depend on each other.

Figure 2: Subscriber Profile Repository Architecture



SPR primarily maintains the subscriber profile information such as Username, Password, Domain, devices configured, services (plans), and so on. SPR database is updated by provisioning process and queried at start of session.

Session Manager (SM)

The session manager database contains all the state information for a call flow. Components such as Diameter or custom components can add additional data to the session database without core code changes.

Multi-Service Balance Manager (MsBM)

MsBM is used to support any use cases that require balance, for example, volume monitoring over Gx also uses the Balance database (without need for Gy). It also handles the CPS implementation of an online charging server (OCS). It handles quota and manages the subscriber accounting balances for CPS. Quota information is stored separately from a subscriber so that it can be shared or combined among multiple subscribers.

MsBM defines the times, rates and balances (quota) which are used as part of CPS. In addition, it performs the following functions:

- Maintains the multiple balances that can be assigned for each subscriber. Balance types can include:
  - Recurring balances (for example, reset daily, monthly, or per billing cycle)
  - One-time balances such as an introductory offer might be assigned per subscriber
  - Both recurring and one time balances can be topped-up (credited) or debited as appropriate
- Balances can be shared among subscribers, as in family or corporate plans.
- Operators can set thresholds to ensure some pre-defined action is taken after a certain amount of quota is utilized. This can be used to prevent bill shock, set roaming caps, and to implement restrictions around family or corporate plans.
- Operators can configure policies to take action when a given threshold is breached. Examples include:

- Sending a notification to the subscriber
- Redirecting the subscriber to a portal or a URL
- Downgrading the subscriber's balance

Note The decision to utilize quota is made on a per service basis. Users who do not have quota-based services would not incur the overhead of querying/updating the MsBM database.

Geographic Redundancy

Overview

CPS can be deployed in a geographic redundant manner in order to provide service across a catastrophic failure, such as data center failure of a site hosting a Policy Suite cluster. In a GR deployment, two Policy Suite clusters are linked together for redundancy purposes with the clusters located either locally or remotely from each other in separate geographic sites.

Geo-redundancy is achieved through data synchronization between the two sites in a geographic redundant pair through a shared persistence layer. The specific subscriber profile, balance, session data replicated across sites is determined based on the deployment architecture, service requirements, and the network environment.

CPS supports active/standby redundancy in which data is replicated from the active to standby cluster. The active site provides services in normal operation. If the active site fails, the standby site becomes the primary and takes over operation of the cluster. In order to achieve a geographically distributed system, two active/standby pairs can be setup where each site is actively processing traffic and acting as backup for the remote site.

CPS also supports active/active deployment model in which data can be replicated across sites in both directions in order to achieve a geographically distributed system. If one system fails, entire traffic would failover to one site that can handle traffic of both sites simultaneously.

Concepts

The following HA/GR concepts and terms are useful in understanding a GR implementation of CPS:

Active/Standby

The Active site is one which is currently processing sessions. The Standby site is idle, waiting to begin processing sessions upon failure of one or more systems at the Active site.

Note Active/Standby and Primary/Secondary are used interchangeably in the context of Active/Standby GR solutions.

Failure

Failure refers to the failure of a given part in functioning. The part may be hardware, software, networking, or other infrastructure (power).

Failover

Failover refers to termination of the application/system at one site and the initiation of the same application/system at another site at the same level. Failovers can be manually triggered, where the system is brought down at the direction of an administrator and restored at a different site, or automatically triggered, in scenarios like, if the master database is not available at primary site without the direction of an administrator.

In both cases, failovers proceed through a set of predefined steps. Manual failover differs from manual intervention wherein depending upon the situation, faults, and so on, there are some additional steps that are executed to make a system Up or Down. Such steps might include patch installations, cleanup, and so on.

Failover Time

Failover Time refers to the duration needed to bring down the system, and start the system at the other site, until it starts performing its tasks. This usually refers to automatic failover.

Heartbeat

Heartbeat is a mechanism in which redundant systems monitor health of each other. If one system detects that the other system is not healthy/available, it can start failover process to start performing the tasks.

Split Brain

Split Brain situation arrives when the link between Primary and Secondary sites goes down, and due to unavailability of Heartbeat response, each site tries to become Primary. Depending upon technologies/solutions used for high availability, the behavior of each site might differ (both becoming Primary or both becoming Secondary and so on.) In general, this is an undesirable situation, and is typically avoided using solutions like Arbiter.

Cross-site Referencing

In a GR deployment, traffic is coming at one or both sites depending upon the nature of deployment. It is required that all the queries to the database be restricted to local sites. However, in case of certain failures, the servers on one site might query databases instances on another site. Since there is a time latency between the two sites, these queries are slow in nature and hence responses get delayed. This is cross-site referencing.

Arbiter

Arbiter is a lightweight 'observer' process that monitors the health of Primary and Secondary systems. Arbiter takes part in the election process of the Primary (active) system. It breaks any ties between systems during the voting process making sure that no split-brain occurs if there is a network partition (network partition is an example). To make sure that this process works smoothly, you can have odd number of participants in the system (for example, Primary, Secondary, and Arbiter).

Data Redundancy

There are two ways to achieve Data Redundancy.

- Data Replication

- Shared Data

Data Replication

In this mechanism, data is replicated between the Primary and Secondary sites so that it is always available to the systems. There are various factors that affect efficiency of replication.

- Bandwidth: Bandwidth is important when the amount of data that is replicated is large. With higher bandwidth, more data can be sent simultaneously. Also, if the data is compressed, it helps further better utilization of bandwidth.
- Latency: Latency is the time required to send a chunk of data from one system to another. The round-trip latency is an important factor that determines speed of replication. Lower latency equates to higher round-trip speed. Latency typically increases with the distance and number of hops.
- Encryption: During replication, the data might travel on public network, it is important to have encryption of data for protection. Encryption involves time and slows the replication. Data needs to be encrypted before it is transmitted for replication which takes additional time.
- Synchronous/Asynchronous: In an asynchronous write and asynchronous replication model, a write to local system is immediately replicated without first waiting for confirmation of the local write. With this form of replication there are chances of data loss if the replication could not take place due to some issue.

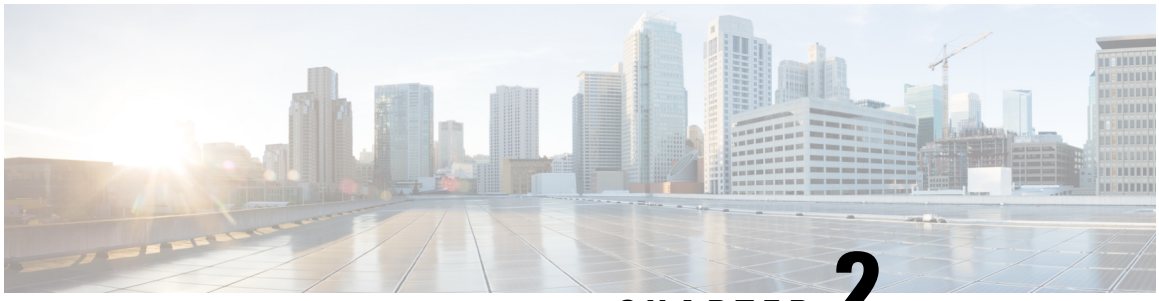
  This risk can be mitigated by the replication system through maintenance of operations log (oplog) which can be used to reconfirm replication. In the combination of asynchronous write and synchronous replication, oplog plays a vital role. The application is made efficient by responding fast to writes and data synchronization can also be ensured.

Shared Data

This is mostly applicable in case of local high availability where the data can be stored on an external shared disk which is not part of the system. This disk is connected to both the systems. In case a system goes down, the data is still available to redundant host. Such a system is difficult to achieve in case of Geographic Redundancy as write time to disk would be significant due to latency.

Operations Log

In the context of MongoDB, the operations log (oplog) is a special capped collection that keeps a rolling record of all the operations that modify the data stored in a database. Operations are applied to the primary database instance which then records the operation in the primary's oplog. The secondary members then copy and apply these operations in an asynchronous process, which allows them to maintain the current state of the database. Whether applied once or multiple times to the target data set, each operation in the oplog produces the same results.



CHAPTER 2

GR Reference Models

- GR Reference Models, on page 9
- Advantages and Disadvantages of GR Models, on page 14
- SPR/Balance Considerations, on page 15
- Data Synchronization, on page 16
- CPS GR Dimensions, on page 17
- Network Diagrams, on page 19

GR Reference Models

The CPS solution stores session data in a document-oriented database. The key advantage is that the application layer responsible for transactional Session data is stored in MongoDB (document-oriented database). Data is replicated to help guarantee data integrity. MongoDB refers to replication configuration as replica sets as opposed to Master/Slave terminology typically used in Relational Database Management Systems (RDBMS).

Replica sets create a group of database nodes that work together to provide the data resilience. There is a primary (the master) and 1..n secondaries (the slaves), distributed across multiple physical hosts.

MongoDB has another concept called Sharding that helps scalability and speed for a cluster. Shards separate the database into indexed sets, which allow for much greater speed for writes, thus improving overall database performance. Sharded databases are often setup so that each shard is a replica set.

The replica set model can be easily extended to a Geo-redundant location by stretching the set across two sites. In those scenarios, an Arbiter node is required. The Arbiter is used as a non-data-processing node that helps decide which node becomes the primary in the case of failure. For example, if there are four nodes: primary, secondary1, secondary2 and the arbiter, and if the primary fails, the remaining nodes “vote” for which of the secondary nodes becomes the primary. Since there are only two secondaries, there would be a tie and failover would not occur. The arbiter solves that problem and “votes” for one node breaking the tie.

Without Session Replication

The following list provides information related to GR without session replication:

- If PCEF elements need to switch over clusters, the current Diameter session between the PCEF and PCRF will be terminated and a new session will need to be re-established.
- Simplifies architecture and reduces complexity.
- Quota data not reported. Currently, this is a limitation.

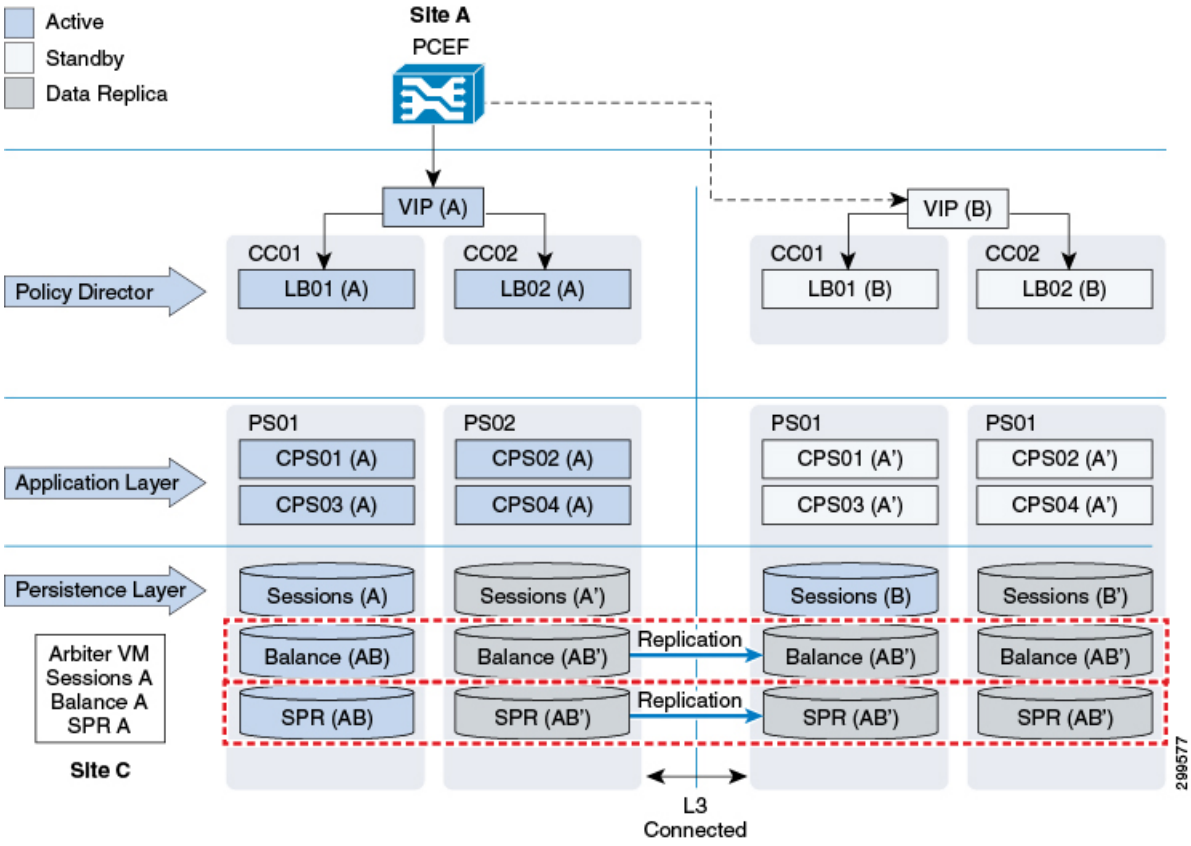
Active/Standby

In active/standby mode, one CPS system is active while the other CPS system, often referred to as the Disaster Recovery (DR) site, is in standby mode. In the event of a complete failure of the primary CPS cluster or the loss of the data center hosting the active CPS site, the standby site takes over as the active CPS cluster. All PCEFs use the active CPS system as primary, and have the standby CPS system configured as secondary.

The backup CPS system is in standby mode; it does not receive any requests from connected PCEFs unless the primary CPS system fails, or in the event of a complete loss of the primary site.

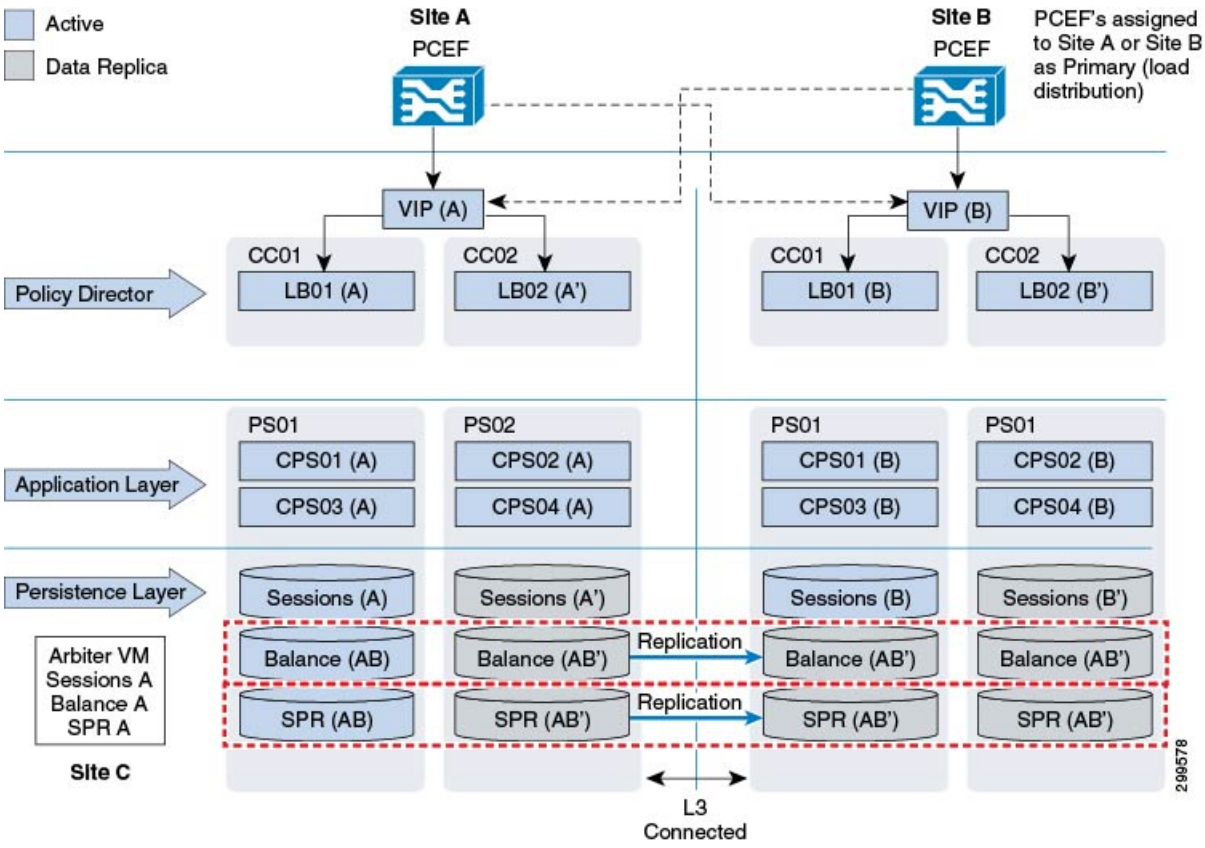
If an external load balancer or Diameter Routing Agent (DRA) is used, the CPS in the active cluster is typically configured in one group and the CPS in the standby cluster is configured in a secondary group. The load balancer/DRA may then be configured to automatically fail over from active to passive cluster.

Figure 3: Active/Standby Without Session Replication



Active/Active

Figure 4: Active/Active Without Session Replication

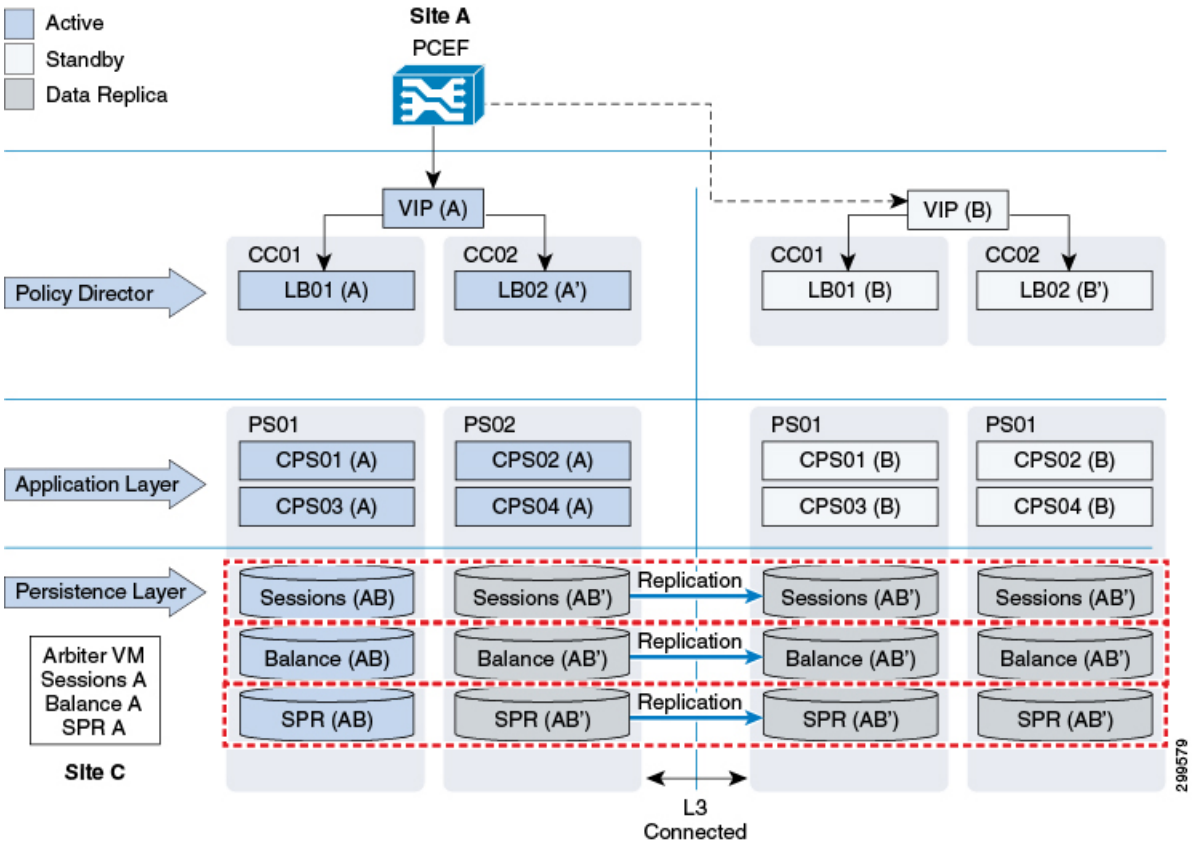


- Traffic from the network is distributed to two CPS clusters concurrently.
- PCEFs are divided within the Service Provider's network to have a 50/50% split based on traffic.
- Session data is not replicated across sites.
- SPR (subscriber information) data is replicated across Standby site.
- Balance data is replicated across Standby site.
- Diameter sessions need to be re-established if a failover occurs. Outstanding balance reservations will time out and be released.
- In case of a failure all traffic is routed to the remaining CPS site.

With Session Replication

Active/Standby

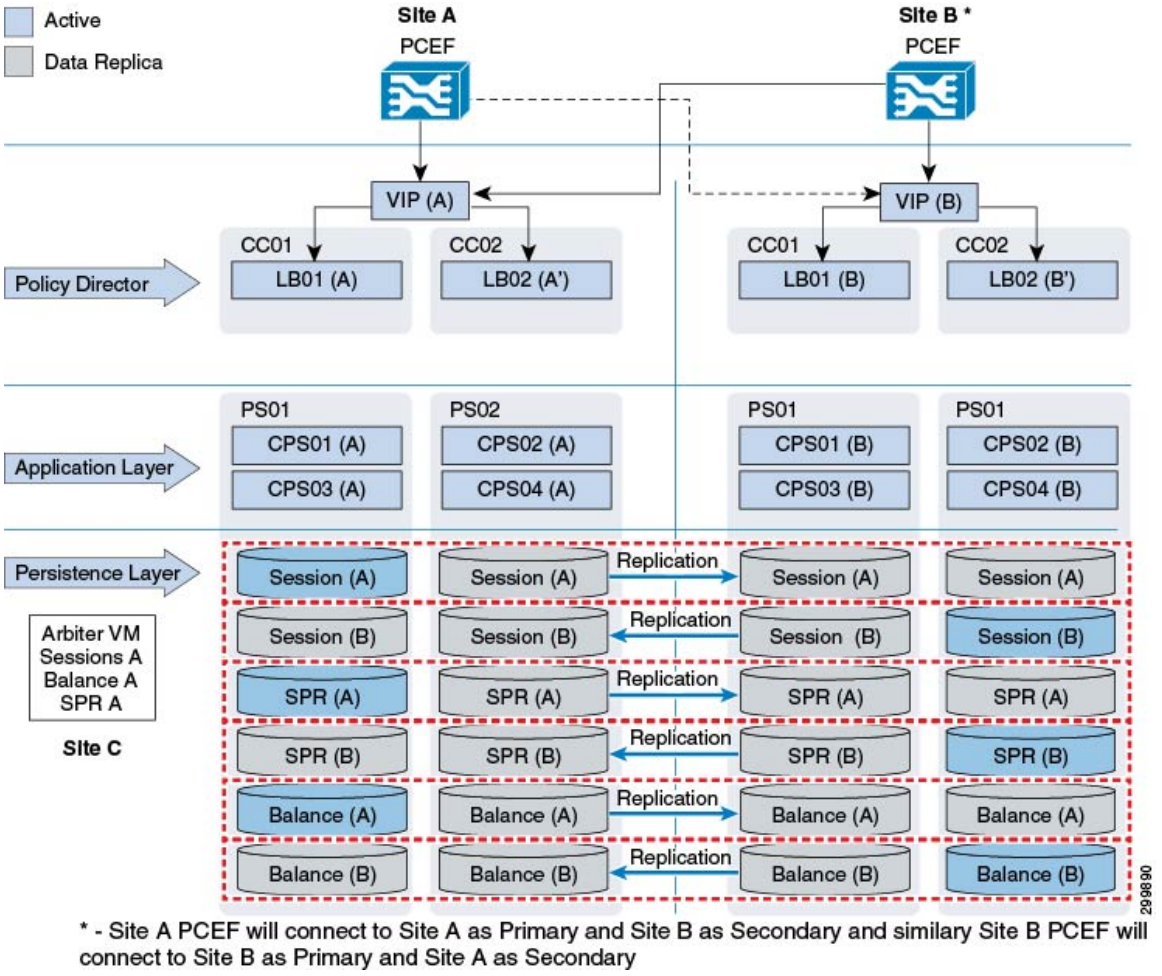
Figure 5: Active/Standby With Session Replication



- Solution protects against complete site outage as well as link failure towards one or more PCEF sites.
- If PCEF fails over to Secondary site while Primary site is still active (for example, link failure):
  - SPR data is retrieved from local SPR replica members at Secondary site.
  - Session and Balance data is read/written across from/to Primary site.
- Complete Outage of Policy Director Layer results in database failover to Secondary site.
- On recovery from a failure, a CPS node does not accept traffic until databases are known to be in a good state.

Active/Active

Figure 6: Active/Active With Session Replication



* - Site A PCEF will connect to Site A as Primary and Site B as Secondary and similary Site B PCEF will connect to Site B as Primary and Site A as Secondary

- Traffic from the network is distributed to two clusters concurrently.
- PCEFs are divided within the Service Provider's network to have a 50/50% split based on traffic.
- Session data is replicated across sites (two way replication).
- SPR (subscriber information) data is replicated across Standby site.
- Balance data is replicated across Standby site.
- Diameter session does not need to be re-established if a failover occurs. No loss of profile or balance information.
- Load balancer VMs use only local VMs for traffic processing.
- In case of a failure all traffic is routed to the remaining site.

Advantages and Disadvantages of GR Models

The following table provides a comparison based on advantages and disadvantages for different GR models described in GR Reference Models, on page 9.

Table 1: Advantages and Disadvantages of GR Models

GR Model	Session	Other Databases	Advantages	Disadvantages
Active/Standby	Replicated	SPR and Balance replicated	Protection against complete site outage as well as link failure towards one or more PCEFs. Session Continuation, diameter sessions do not need to be re-established, hence VoLTE friendly.	Session replication demands bandwidth. In case there is network latency or high TPS, the hardware requirement increases as we are required to split the incoming traffic across multiple virtual machines to achieve high speed replication and recovery.
Active/Standby	Not replicated	SPR and Balance replicated	Protection against complete site outage as well as link failure towards one or more PCEFs.	Sessions do not continue after failover, hence, they need to be re-established, NOT VoLTE friendly.
Active/Active	Replicated	SPR and Balance replicated, they are separate to each site	Protection against complete site outage as well as link failure towards one or more PCEFs. Session Continuation, diameter sessions do not need to be re-established, hence VoLTE friendly.	Session replication demands bandwidth. The hardware requirement increases significantly as we need additional load balancers and session cache virtual machines.

GR Model	Session	Other Databases	Advantages	Disadvantages
Active/Active	Not replicated	SPR and Balance replicated, they are separate to each site	Protection against complete site outage as well as link failure towards one or more PCEFs. Low bandwidth and significantly low hardware requirements.	Sessions do not continue after failover, hence, they need to be re-established, NOT VoLTE friendly.

SPR/Balance Considerations

SPR Considerations

The following list provides the information related to SPR considerations:

- SPR data is read from secondary replica members:
  - MongoDB tag sets can be used to target read operations to local replica members, that way avoiding cross-site SPR reads.
- SPR data is always written to primary database:
  - Profile updates are broadcasted to other sites to trigger policy updates if/as required for sessions established through remote sites.
- SPR updates that happen while primary site is isolated are only enforced after session update (once primary site is available again).

Balance Considerations

The following list provides the information related to balance considerations:

- Balance data is always read from primary database unless primary database is not available (for example, in the event of site isolation).
- Balance data is always written to primary database.
- Balance database design options:
  - Single database across two GR sites: cross-site balance read/writes from site without primary database.
- CDR Balance Records Reconciliation
  - During site isolation, debits are written to backup CDR balance database for reconciliation when connectivity is restored.

- No thresholds or caps are enforced during site isolation.
- Policies associated with any threshold breaches during isolation are enforced at the time of balance reconciliation.
- Potential for balance leakage if balance consumed during isolation is greater than user‘s remaining allowance.

Data Synchronization

Geo-redundancy is achieved by synchronizing data across the site(s) in the cluster. Three types of data are replicated across sites:

- Service and policy rule configuration
- Subscriber data is stored in the SPR component
- Balance data stored in the MsBM component

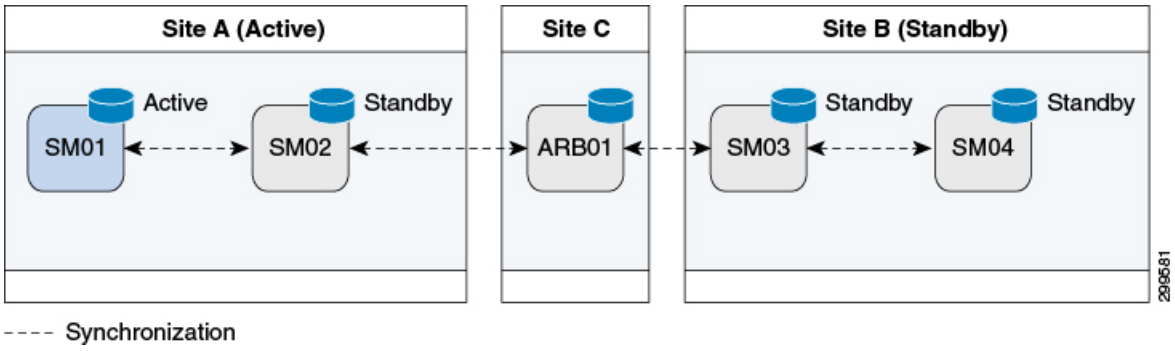
In addition, active session data stored in the Session Manager component may also be synchronized across sites when network conditions permit. Active session data is the most volatile data in CPS and has the most stringent synchronization requirements.

CPS utilizes a unicast heartbeat between sites in the geographic redundant solution. The heartbeat allows the session manager components to know which is the currently active component and protects against a split-brain scenario where data is accepted at more than one session manager component (possibly causing data corruption).

An additional external component called an “arbiter” provides a tie-breaking vote as to which of the session managers is the current master. This external component is required to reside on a separate site from the primary and secondary sites and must be routable from both sites. This is used to ensure that if one of the sites is lost, the arbiter has the ability to promote the standby sites session manager to be the master.

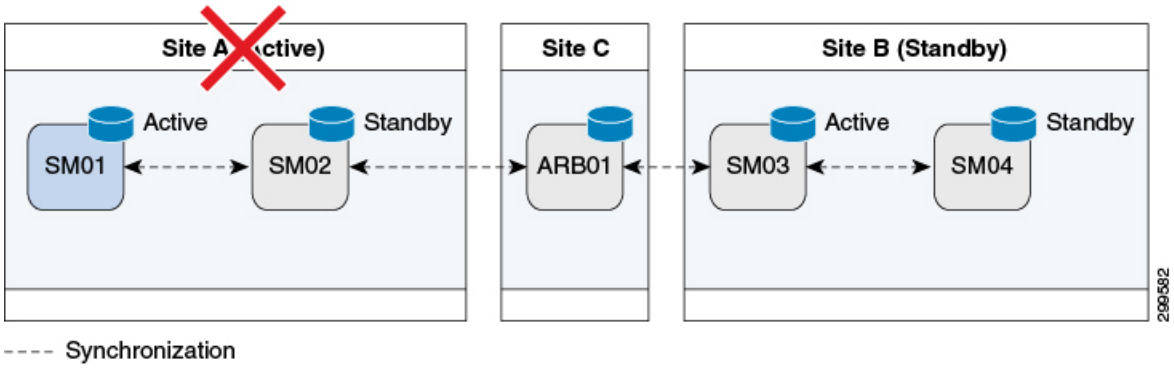
The following example shows a detailed architecture of the data synchronization for subscriber, balance and session data:

Figure 7: Data Synchronization for Subscriber, Balance and Session Data



In the case of Site A failure, Site B's session manager will become master as shown in the following example:

Figure 8: In Case of Site A Failure



Data Synchronization in MongoDB

In short, replication is achieved through a replica set where there are multiple members of a set. These members have only one primary member and others are secondary members. Write operations can occur only in primary, and read operations can happen from Primary and Secondary members. All the data written to Primary is stored in form of operation logs, that is, oplogs on primary database and secondaries fetch that to synchronize and remain up to date. In CPS, `/etc/broadhop/mongoConfig.cfg` file defines replica members, replica sets and therefore, defines databases that will be replicated and not replicated.

For more information on data synchronization in MongoDB, refer to http://docs.mongodb.org/manual/core/replica-set-sync/

CPS GR Dimensions

The GR dimensions such as databases, number of sites, arbiter considerations, and shards are dependent on each other. Only the deployment style is not dependent on other dimensions. Rest of the dimensions are inter-related and,

- Arbiter typically decides models for shard.
- Number of sites impacts the decision to have database in common.
- Common database style impacts decision to have shard.

Different Databases

CPS has three databases that have subscriber critical data such as subscriber database (SPR), balance, and sessions. Different deployment models exist depending upon how the user wants to have the databases configured. Some users might want a database common across different sites (typically this can happen for SPR), or individual instances at each site (most of the times this would be with sessions database and balance database). Typically, the databases that are updated more frequently (such as sessions and balance) would be maintained locally and replicated across sites whereas databases that are updated rarely can be kept common across sites (with some limitations).

Number of Sites

Typical deployment is expected to be two sites. However, there might be cases where multiple combinations might come up with respect to database redundancy, common database across multiple sites, general redundancy across multiple sites and so on. Since this is a highly variable factor, for each deployment model here, we need to understand various network requirements.

Arbiter Considerations

Typically the Arbiter needs to be located at a third independent site. However, depending upon customer needs and limitations, different deployment models come up where arbiter can be placed at one of the sites, creating limitations in the model.

The location of the Arbiter is an important factor in the design. Having the Arbiter located on the same site as that of Primary or Secondary poses various issues. The following table describes the issues:

Table 2: Issues Posed by Location of the Arbiter

Distribution of Arbiter	Impact on System
Arbiter at Active site	When Active site goes down, database on Secondary is supposed to become Primary. However, since it does not have required votes as Arbiter is also down at Primary site, role change cannot take place and we face downtime.
Arbiter on Secondary site	In this case, if Secondary site goes down, we do not have arbiter available. Due to this, database on Primary site does not have majority of votes, and database steps down. That way, we face downtime on system unless there is manual intervention. Additionally, if there is a split brain situation, since arbiter is on secondary site, database role changeover starts from Primary to Secondary, which is unnecessary.
Arbiter on third site	This is the best and recommended way of placing an arbiter. In any case, either Primary failure or Secondary failure, a proper failover happens as there are always majority of votes available to select a Primary.

It is important to understand the placement of arbiter and its implications. In Geographic Redundancy, failover is expected when a site goes down completely. There are many possibilities for a site to go down and based on these possibilities, we can decide the location of arbiter.

Database Partitioning - Shards

When the database size grows large, it is good to have it partitioned, in terms of MongoDB. The partitioning is done by creating shards for the database. MongoDB has some limitations for creating shards and depending upon deployment model, shard considerations come in picture. When shards come in picture, we need to also

consider the configuration servers for those shards. The configuration server decides which partition/shard contains what data. It has the keys based on which data distribution and lookup works.

Placement of these configuration servers also plays an important role in performance of databases. During site failures, if we have less number of configuration servers available, the performance of database is degraded. Hence, it is important to place the configuration servers in such a way that maximum of them are available always. Typically, the configuration servers are placed in line with database that is, one at primary, another at secondary and third at the arbiter. MongoDB supports a maximum of three configuration servers.

Session Shard Considerations

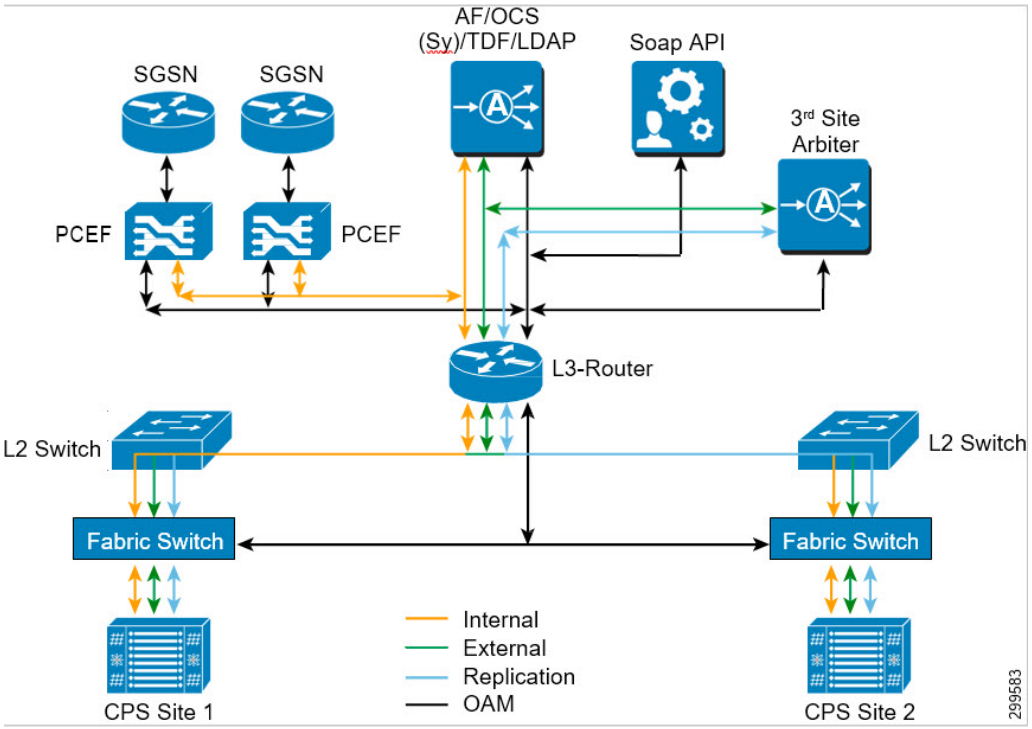
For sessions, we define internal shards. Currently, we create four internal shards per session database so that we see four internal databases. This helps to achieve parallel writes to same database thereby increasing write/read efficiency and achieve higher performance. Typically, for higher TPS, we might be required to create multiple shards across different virtual machines. In that case, an additional session replica set is created that contains four more shards. The admin database contains information for all such shards so that the Policy Server (QNS) processing engines route session calls to appropriate shards based on internal hashing algorithm. The actual number of shards required can be obtained from the dimensioning guide.

Network Diagrams

High Level Diagram including other Network Nodes

The following is an example of a high-level diagram showing various network modules connected to CPS for GR setup. This diagram can be different for different deployment scenarios. Contact your Cisco Technical Representative for high level diagram specific to your deployment scenario.

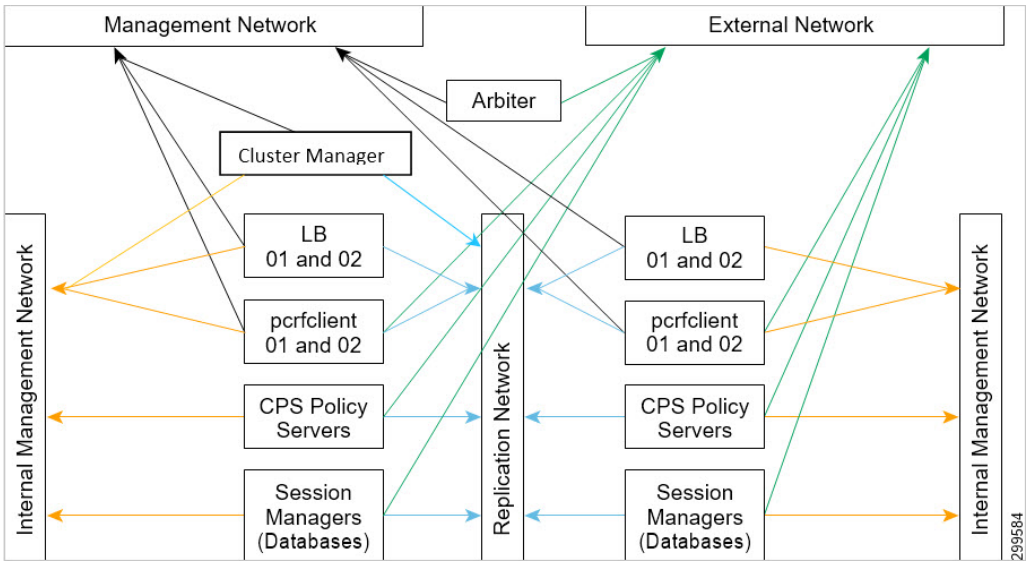
Figure 9: High Level Diagram including other Network Nodes



CPS Level Network Diagram

The following network diagram explains various requirements for GR setup:

Figure 10: CPS Level Network Diagram



The following sections describe the interface requirements for GR setup:

Management Network Interface

This interface is used for traffic to CPS, unified API, portal (not shown here), and for login to CPS machines through pcrfclient, and to access Policy Builder and Control Center web interfaces.

The following VMs need this network interface:

- Load balancers
- pcrfclient01
- Arbiter
- Cluster Manager

External and Routable Network Interface

This interface is used for communication between with any entity that is external to CPS HA System. Since the MongoDB configuration servers reside on pcrfclient01 of both sites, a separate network is needed for both to communicate with each other over an interface other than replication network. If the replication network fails, communication would still be needed between the arbiter and session managers, and between arbiter and pcrfclient01 so that the arbiter is able to determine the appropriate primary for databases, and make more than one configuration servers available. If this is not done, and if the arbiter is configured to communicate with databases over the replication network, if the replication network fails, a split brain situation occurs since the arbiter would be disconnected from both sites.

When there are no shards configured for databases, no configuration servers are needed. The pcrfclient01 at both sites still needs external network connectivity with arbiter as scripts on pcrfclient need to communicate with the arbiter (such as `get_replica_status.sh`).

In GR, we need to connect the Policy Server (QNS) to arbiter. During failover, Policy Server (QNS) gets all the available members' list and tries to see if they are reachable. In case arbiter is not reachable from Policy Server (QNS), it hangs there.

The following VMs need this network interface:

- pcrfclient01
- Arbiter
- Session managers
- Policy Server (QNS)

Replication Network Interface

Typically referred as Signaling Network, this network carries the data replication in a Geo-HA environment across two sites. Also, Policy Servers (QNS) on one site communicate with databases on another site using the same interface. The same network should be used to exchange messages between two sites.

The following VMs need this network interface:

- Policy Director (lbs)
- pcrfclient

- Policy Server (QNS)
- Session managers (databases)
- Cluster Manager

Note In Geo-HA environment, if you want to execute the platform scripts (such as, `diagnostics.sh`) from Cluster Manager, it must be connected to other site's Session Manager VM. Hence, Cluster Manager must be a part of external or replication network.

Internal Network

This network is used for internal communication of virtual machines of the same site.

All the CPS VMs need this network interface.

Summary

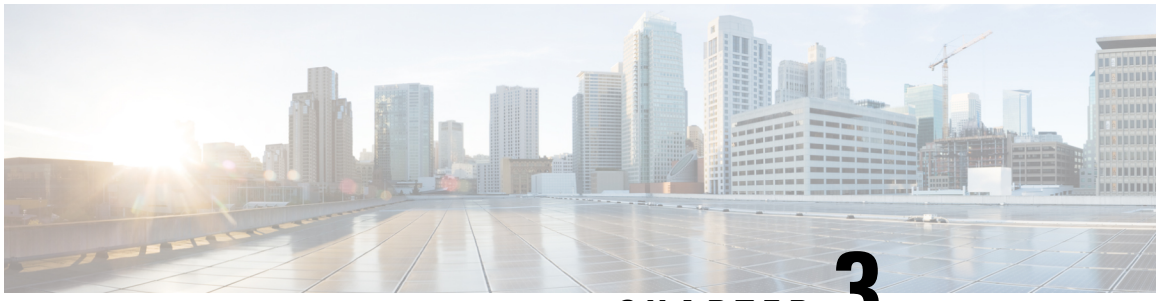
The following table provides a summary of the different VM network requirements:

Table 3: Summary

VM Name	Management IP	Signaling IP/Replication	Internal IP	External Non-Management IP
Cluster Manager	Yes	Yes	Yes	No
pcrfclient01/02	Yes	Yes	Yes	Yes
lb01/lb02	Yes	Yes	Yes	No
qns01-n	No	Yes	Yes	Yes
sessionmgrs	No	Yes	Yes	Yes
arbiter	Yes	No	No	Yes

Network Requirements

Bandwidth and latency are to be obtained from Cisco Technical Representative depending upon your deployment model.



CHAPTER 3

GR Installation - VMware

- GR Installation Process, on page 23
- Overview, on page 23
- Prerequisites, on page 24
- Reference for CPS VM and Host Name Nomenclature, on page 25
- Arbiter Installation, on page 27
- Standalone Arbiter Deployment On VMware, on page 30
- Configure Remote/Peer Site VM, on page 32
- Database Configuration, on page 34
- Balance Backup Database Configuration, on page 37
- Session Cache Hot Standby, on page 40
- Policy Builder Configuration, on page 43
- Access Policy Builder from Standby Site when Primary Site is Down, on page 46
- qns.conf Configuration Changes for Session Replication, on page 47
- Configurations to Handle Database Failover when Switching Traffic to Standby Site Due to Load Balancer Fail/Down, on page 48

GR Installation Process

In this chapter, Active/Standby Geographic Redundancy model has been used to describe the database and configuration changes required to modify the current installed HA system into Geo-HA.

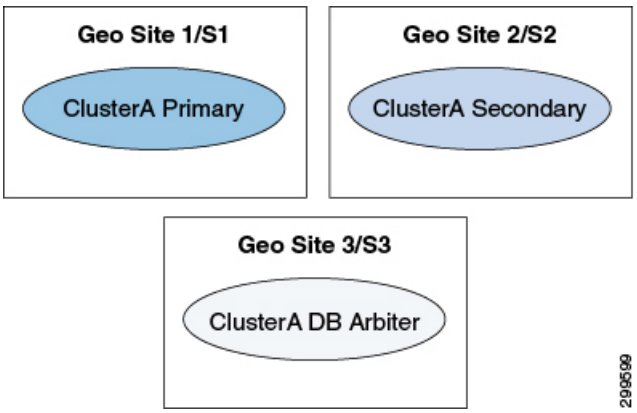
If you want to deploy historical Active/Active model, just deploy additional flipped pair of this active/standby model.

Overview

An overview of active/standby model has been provided in this section.

1. Active/Standby solution has only one CPS cluster at each site, CA-PRI (referenced as ClusterA Primary henceforth) at S1 site (referenced as Geo-site-1/site-1 henceforth) and CA-SEC (referenced as ClusterA secondary henceforth) at S2 site (referenced as Geo-site-2/site-2 henceforth).

Figure 11: Geographical Sites



In the above figure, you have primary cluster (Geo Site 1/S1), secondary cluster (Geo Site 2/S2) and arbiter (Geo Site 3/S3).

- Geo site 1/S1 could be any site (for example, Mumbai)
- Geo site 2/S2 could be any site (for example, Chennai)
- Geo site 3/S3 could be any site (for example, Kolkata)

2. For Site1 PCEF, there are two CPS clusters. One is primary, CA-PRI on S1 and other is secondary, CA-SEC on S2. They are geographically redundant.
3. Upon failure of primary CPS Cluster, secondary CPS cluster would seamlessly serve the subscriber's sessions. For that, session replication is enabled between primary and secondary clusters and for session replication high bandwidth is expected. For more information, contact your Cisco technical representative.
4. We recommend to use Replication interface for Database replication between Geo sites (that is, S1 and S2) to segregate Network traffic with Database replication traffic. For example, setting up separate VLAN's for segregating Network and Database traffic.
5. The secondary CPS cluster is not configured as passive.
6. We recommend to place the arbiter on site-3.
7. We recommend the SPR and balance databases to be on SSD and session database to be on tmpfs for optimized performance.

Prerequisites

- Base install (CPS-HA) has been completed on both sites and verified basic validation on both sites.
- Call model has been validated on both HA sites as per your TPS/traffic.
- CPS VMs should have Replication IP address.
- Familiarity with *CPS Installation Guide for VMware*.

  Familiarity with *CPS Release Notes*.

- For third site, Arbiter must be deployed and running the same build (The ISO used to prepare the Geo-Redundant Setup).
- The database configuration is planned.

Reference for CPS VM and Host Name Nomenclature

Note This section is for reference only. You need to follow the nomenclature based on your network requirements. As a prerequisite, HA must be already deployed.

For better usability of the system, install the HA system according to the following nomenclature:

1. In order to know the exact geo site details, we recommend to have the following entries in VMSpecification sheet of CPS_deployment_config_template.xlsm or VMSpecification.csv.

   Host Name Prefix field value as Sx:

   Table 4: Host Name Prefix Example

Cluster Name	Recommended Value
CA-PRI	S1-
CA-SEC	S2-

2. In order to know the exact cluster name and role (primary/secondary) details, we recommend to have the following entries in Hosts sheet of CPS_deployment_config_template.xlsm or Hosts.csv:
   - Guest Name field value as:

     CA-PRI-XXX for primary cluster (like CA-PRI-lb01, CA-PRI-qns01, and so on.) and CA-SEC-XXX for secondary cluster (like CA-SEC-qsn01, CA-SEC-lb01, and so on.)

3. We recommend to distribute session manager VMs equally between primary and secondary clusters, example:

   sessionmgr01, sessionmgr02, sessionmgr03, sessionmgr04 on CA-PRI and

   sessionmgr01, sessionmgr02, sessionmgr03, sessionmgr04 on CA-SEC

4. The following convention must be used while creating cross site replica-set for the session database:

   You must create the session database replica-set members on same VM and same port on both sites. For example, among four replica-set members (except arbiter), if `sessionmgr01:27717` and `sessionmgr02:27717` are two members of replica-set from SITE1 then choose `sessionmgr01:27717` and `sessionmgr02:27717` of SITE2 as other two replica-set members as shown in following example:

```
[SESSION-SET]
         SETNAME=set01
         OPLOG_SIZE=5120
         ARBITER=SITE-ARB-sessionmgr05:27717
         ARBITER_DATA_PATH=/var/data/sessions.1/set1
         PRIMARY-MEMBERS
         MEMBER1=SITE1-sessionmgr01:27717
```

```
MEMBER2=SITE1-sessionmgr02:27717
SECONDARY-MEMBERS
MEMBER1=SITE2-sessionmgr01:27717
MEMBER2=SITE2-sessionmgr02:27717
DATA_PATH=/var/data/sessions.1/set1
[SESSION-SET-END]
```

5. pcrfclient01 and pcrfclient02 of each site require Management/Public IP
6. Site1 HA Blade naming conventions of VMs looks like (This information is for reference only):

Table 5: Naming Convention

Blade	Virtual Machines
CC Blade 1	S1-CA-PRI-cm S1-CA-PRI-lb01 S1-CA-PRI-pcrfclient01
CC Blade 2	S1-CA-PRI-lb02 S1-CA-PRI-pcrfclient02
CPS Blade 1	S1-CA-PRI-qns01 S1-CA-PRI-sessionmgr01
CPS Blade 2	S1-CA-PRI-qns02 S1-CA-PRI-sessionmgr02
CPS Blade 3	S1-CA-PRI-qns03 S1-CA-PRI-sessionmgr03
CPS Blade 4	S1-CA-PRI-qns04 S1-CA-PRI-sessionmgr04

7. Site2 HA configuration looks like (This information is for reference only):

Table 6: Naming Convention

Blade	Virtual Machines
CC Blade 1	S1-CA-SEC-cm S1-CA-SEC-lb01 S1-CA-SEC-pcrfclient01
CC Blade 2	S1-CA-SEC-lb02 S1-CA-SEC-pcrfclient02
CPS Blade 1	S1-CA-SEC-qns01 S1-CA-SEC-sessionmgr01

Blade	Virtual Machines
CPS Blade 2	S1-CA-SEC-qns02 S1-CA-SEC-sessionmgr02
CPS Blade 3	S1-CA-SEC-qns03 S1-CA-SEC-sessionmgr03
CPS Blade 4	S1-CA-SEC-qns04 S1-CA-SEC-sessionmgr04

Arbiter Installation

Note If you want to add the MongoDB authentication on Arbiter, refer to *General Configuration* section in *CPS Installation Guide for VMware*. You need to mention password for all the sites separately using CSV file and that should be same for all the sites.

On Third Site

Important Currently, SNMP and statistics are not supported on third site arbiter.

Do not install Arbiter if third site is not there or Arbiter is already installed on primary site.

Additionally, if third site blades are accessible from one of the GR sites, you can spawn the Arbiter VM from one of the sites, say, Site1, and installer will sit on third site blades. In that case also, this section is not applicable. Just have appropriate configurations done (On Primary Site, on page 29) so that destination VM is on a third site's blade.

The automatic GR site failover happens only when arbiters are placed on third site thus we recommend the MongoDB arbiter to be on third site that is, S3.

Note Arbiter VM name should be sessionmgr*xx*.

Site3 HA configuration looks like (This information is for reference only):

Table 7: Naming Convention

Blade	Virtual Machines	vCPU	Memory (GB)
CPS Blade 1	S3-ARB-cm S3-CA-ARB-sessionmgr01	1 4	8 8

For more information about deploying VMs, refer to *CPS Installation Guide for VMware*.

Step 1 Configure system parameters for deployment for new Arbiter VM. We need the following CSV files to deploy and configure arbiter VM:

They are:

- VLANs.csv
- Configuration.csv
- VMSpecification.csv
- AdditionalHosts.csv
- Hosts.csv

1. VLAN.csv: Here configurations need to be as per targeted deployment/availability. An example configuration is shown:

Table 8: VLAN.csv

VLAN Name	Network Target Name	Netmask	Gateway	VIP Alias
Internal	VM Network	x.x.x.x	x.x.x.x	-
Management	VM Network-1	x.x.x.x	x.x.x.x	-

2. Configuration.csv: Here configurations need to be as per targeted deployment/availability.
3. VMSpecification.csv: Here configurations need to be as per targeted deployment/availability.
4. AdditionalHosts.csv: Here configurations need to be as per targeted deployment/availability. An example configuration is shown where we need to provide site1 and site2 session managers details:

Table 9: AdditionalHosts.csv

Host	Alias	IP Address
ntp-primary	ntp	x.x.x.x
ntp-secondary	btp	x.x.x.x
CA-PRI-sessionmgr01	-	x.x.x.x
CA-PRI-sessionmgr02	-	x.x.x.x
CA-SEC-sessionmgr01	-	x.x.x.x
CA-SEC-sessionmgr02	-	x.x.x.x

5. Hosts.csv: Take the template file `/var/qps/install/current/scripts/deployer/templates` from Cluster Manager VM and make changes.

   An example configuration is shown:

Figure 12: Hosts.csv

Hypervisor Name	Guest Name	Role	Alias	Datastore	Networks -->	Internal	Management Gx
x.x.x.x	CA-ARB-sessionmgr01	smarb	sessionmgr01	x.x.x.x		x.x.x.x	

299596

Note
- Datastore name should be as per deployment/availability.
- Arbiter VM alias should be sessionmgr*XX* only. In the above example, it is sessionmgr01.

Step 2 Convert the Excel spreadsheet into a CSV file and upload the file to the Cluster Manager VM in `/var/qps/config/deploy/csv` directory.

a) Execute the following commands to import CSV files and conversion to JSON data:

```
/var/qps/install/current/scripts/import/import_deploy.sh
```

b) Execute the following command to validate the imported data:

```
cd /var/qps/install/current/scripts/deployer/support/
python jvalidate.py
```

The above script validates the parameters in the Excel/csv file against the ESX servers to make sure ESX server can support the configuration and deploy VMs.

Step 3 For each host that is defined in the Hosts sheet of the deployment spreadsheet, perform the manual deployment (Refer to the *Manual Deployment* section in the *CPS Installation Guide for VMware*).

Example:

An example is shown below:

```
/var/qps/install/current/scripts/deployer
./deploy.sh sessionmgr01
```

Step 4 If you want to enable mongo authentication on Arbiter, create the file `/etc/facter/facts.d/mongo_auth.txt` using the following data. The password should be same with all other sites.

```
db_authentication_enabled=TRUE
db_authentication_admin_passwd=XXXXX
db_authentication_readonly_passwd=YYYYY
```

where, *XXXXX* and *YYYYY* are encrypted passwords.

On Primary Site

Note Optional: Do not perform the following steps if Arbiter is installed on third site.

If third site is not available then deploy arbiter VM on Primary Cluster that is, CA-PRI.

Note Arbiter VM name should be sessionmgr*XX* only. *XX* should be replaced with a digit higher than the last used digit of the session manager. For example, if there are a total of six sessionmgrs (sessionmgr01-sessionmgr06) then, the Arbiter session manager must be sessionmgr07.

To deploy arbiter on primary site, perform the following steps:

Step 1 Configure system parameters for deployment.

Add the following arbiter entry in **Hosts** sheet of deployment template sheet or **Hosts.csv** file. An example entry is shown below m:

Figure 13: Arbiter Entries

Hypervisor Name	Guest Name	Role	Alias	Datastore	Networks -->	Internal	Management Gx	Replication
x.x.x.x	CA-PRI-sessionmgr07	smarb	sessionmgr07	datastore1		x.x.x.x		x.x.x.x

299597

Step 2 Import modified CSV files using the following command:

```
/var/qps/install/current/scripts/import/import_deploy.sh
```

Step 3 Execute the following command to validate the imported data:

```
cd /var/qps/install/current/scripts/deployer/support/
```

```
python jvalidate.py
```

Note The above script validates the parameters in the Excel/csv file against the ESX servers to make sure ESX server can support the configuration and deploy VMs.

Step 4 For each host that is defined in the Hosts sheet of the excel document perform the manual deployment (Refer to the *Manual Deployment* section in the *CPS Installation Guide for VMware*).

An example is shown below:

```
/var/qps/install/current/scripts/deployer
```

```
./deploy.sh sessionmgr07
```

Standalone Arbiter Deployment On VMware

Note If you want to add the MongoDB authentication on Arbiter, refer to *General Configuration* section in *CPS Installation Guide for VMware*. You need to mention password for all the sites separately using CSV file and that must be same for all the sites.

To install Arbiter on VM, perform the following steps:

Step 1 Convert the Cluster Manager VM to an Arbiter (VMware).

Note Here you are converting the Cluster Manager deployed at Site3 to an Arbiter. For more information on how to deploy Cluster Manager VM, refer to *Deploy the Cluster Manager VM* section in the *CPS Installation Guide for VMware*.

Step 2 Run `install.sh` from ISO directory.

```
cd /mnt/iso
./install.sh
Please enter install type [mobile|wifi|mog|pats|arbiter|dra|andsf|escef]: arbiter ----> Select arbiter
 for this option
Would you like to initialize the environment... [y|n]: y     ----> Enter y to continue
```

Note RADIUS-based policy control is no longer supported in CPS 14.0.0 and later releases as 3GPP Gx Diameter interface has become the industry-standard policy control interface.

Note Currently, eSCEF is an EFT product and is for Lab Use Only. This means it is not supported by Cisco TAC and cannot be used in a production network. The features in the EFT are subject to change at the sole discretion of Cisco.

Step 3 When prompted for `Please pick an option for this setup:`.

Select **1** for new Arbiter deployment.

Step 4 To enable the firewall, it is required to add the following configuration in `/etc/facter/facts.d/qps_firewall.txt` file:

```
firewall_disabled=0
internal_address=XX.XX.XX.XX ---> update XX.XX.XX.XX to your internal IP address
internal_device=0            ---> update 0 to your device ID
internal_guest_nic=eth0     ---> update eth0 to other port if it is not using default NIC for internal
 address
```

Step 5 When `install.sh` finishes its run, execute the `reinit.sh` script to apply the appropriate configurations to the system:

```
/var/qps/install/current/scripts/upgrade/reinit.sh
```

Step 6 If you want to enable mongo authentication on Arbiter, create the file `/etc/facter/facts.d/mongo_auth.txt` using the following data. The password should be same with all other sites.

```
db_authentication_enabled=TRUE
db_authentication_admin_passwd=XXXXX
db_authentication_readonly_passwd=YYYYY
```

where, *XXXXX* and *YYYYY* are encrypted passwords. For encrypted passwords, you need to SSH to a Cluster Manager and execute the following command:

```
/var/qps/bin/support/mongo/encrypt_passwd.sh <Password>
```

Step 7 After performing the upgrade/new installation, unmount the ISO image. This prevents any “device is busy” errors when a subsequent upgrade/new installation is performed.

```
cd /root

umount /mnt/iso
```

Step 8 (Optional) After unmounting the ISO, delete the ISO image to free the system space.

`rm` *xxxx.iso*

where, *xxxx.iso* is the name of the ISO image used.

Step 9 (Optional) Change the host name of the Arbiter.

a) Run `hostname` *xxx*, where *xxx* is the new host name for the Arbiter.
b) Edit `/etc/sysconfig/network` to add the new host name for the Arbiter.

Configure Remote/Peer Site VM

Session Manager VM Information on Local Site

Note The following steps need to be performed on other sites as well.

Note In this section, to configure remote/peer site VM in local site, sessionmgr has been taken as an example. You can use this section to add peer policy server (qns) and peer policy directors (lbs) entries in `AdditionalHosts` file.

Step 1 Add the following entry in `AdditionalHosts` sheet of CPS deployment template or `AdditionalHosts.csv` on CA-PRI-cm: Objective of this section is for primary cluster to add other cluster (that is, secondary cluster) session manager's details.

a) Add sessionmgr VM information of secondary cluster (that is, Name, Replication Interface IP addresses).
b) In Alias column add `psessionmgr`*xx* (that is, peer sessionmgr).
c) Add arbiter VM entry, also in Alias column add the same host name.
   - If it is on third site, then add IP address of arbiter VM which is reachable from all other sessionmgrs from both sites.
   - Else add internal interface IP address of arbiter VM.

Example:

Example of `/var/qps/config/deploy/csv/AdditionalHosts.csv` (on CA-PRI-cm):

```
Host,Alias,IP Address
-----
CA-SEC-sessionmgr01,psessionmgr01,xx.xx.xx.xx
CA-SEC-sessionmgr02,psessionmgr02, xx.xx.xx.xx
CA-ARB-sessionmgr01,CA-ARB-sessionmgr01,xx.xx.xx.xx
-----
```

Step 2 Import modified CSV files by executing the following command:

```
/var/qps/install/current/scripts/import/import_deploy.sh
```

Step 3 Execute the following command to validate the imported data:

```
cd /var/qps/install/current/scripts/deployer/support/

python jvalidate.py
```

Note The above script validates the parameters in the Excel/csv file against the ESX servers to make sure ESX server can support the configuration and deploy VMs.

Step 4 Execute the following command in Cluster Manager to copy updated `/etc/hosts` file to all deployed VMs:

```
SSHUSER_PREFERROOT=true copytoall.sh /etc/hosts /etc/hosts
```

Step 5 Validate setup using `diagnostics.sh` script.

Policy Director (lb) VM Information on Local Site

Before you begin

Redis must be enabled as IPC. For more information on how to enable REDIS, refer to *CPS Installation Guide for VMware*.

Step 1 Add the following entry in `AdditionalHosts` sheet of CPS deployment template or `AdditionalHosts.csv` on CA-Site1-cm: Objective of this section is for local site to add other site (that is, remote clusters) policy director (lb) VM details.

a) Add policy director (lb) VM information of secondary cluster (that is, Name, Policy Director (LB) External Interface Name).

b) In Alias column `add plbxx` (that is, peer policy director (lb). For example, plb01, plb02 and so on).

Add IP address of remote policy director (lb) VM which is reachable from all policy director (lb) VMs of primary cluster.

Example:

Example of `/var/qps/config/deploy/csv/AdditionalHosts.csv` (on CA- Site1-cm):

```
Host,Alias,IP Address
-----
CA- Site2-lb01,plb01,xx.xx.xx.xx
CA- Site2-lb02,plb02, xx.xx.xx.xx
-----
```

Step 2 Add the number of remote redis instances in `Configuration.csv` with key as *remote_redis_server_count* and value as the number of redis instances running on remote site:

Example:

If the remote site contains three redis instances per policy director (lb), add the following:

```
remote_redis_server_count,3
```

For more information in *remote_redis_server_count*, refer to *CPS Installation Guide for VMware*.

Step 3 Import modified CSV files by executing the following command:

```
/var/qps/install/current/scripts/import/import_deploy.sh
```

Step 4 Execute the following commands in Cluster Manager to copy updated `/etc/hosts` file to all deployed VMs:

```
SSHUSER_PREFERROOT=true copytoall.sh /etc/hosts /etc/hosts

copytoall.sh /etc/broadhop/redisTopology.ini /etc/broadhop/redisTopology.ini
```

Step 5 Verify that the `/etc/broadhop/redisTopology.ini` contains the remote policy director (lb) redis instances entry as follows:

```
cat /etc/broadhop/redisTopology.ini

policy.redis.qserver.1=lb01:6379
policy.redis.qserver.2=lb01:6380
policy.redis.qserver.3=lb01:6381
policy.redis.qserver.4=lb02:6379
policy.redis.qserver.5=lb02:6380
policy.redis.qserver.6=lb02:6381
remote.policy.redis.qserver.1=plb01:6379
remote.policy.redis.qserver.2=plb01:6380
remote.policy.redis.qserver.3=plb01:6381
remote.policy.redis.qserver.4=plb02:6379
remote.policy.redis.qserver.5=plb02:6380
remote.policy.redis.qserver.6=plb02:6381
```

If the number of redis instances/lb instances are to be increased/decreased on the remote cluster(s), the same should first be updated in all other clusters in the CSV files as mentioned in Step 1, on page 33 and Step 2, on page 33.

Repeat the steps from Step 3, on page 34 to Step 5, on page 34 after changing the CSV files so as to update the `redisTopology` file on all the VMs.

Database Configuration

Note While configuring mongo ports in a GR environment, there should be a difference of 100 ports between two respective sites. For example, consider there are two sites: Site1 and Site2. For Site1, if the port number used is 27717, then you can configure 27817 as the port number for Site2. This is helpful to identify a mongo member's site. By looking at first three digits, one can decide where the mongo member belongs to. However, this is just a guideline. You should avoid having mongo ports of two different sites close to each other (for example, 27717 on Site-1 and 27718 on Site2).

Reason: The reason is that the `build_set.sh` script fails when you create shards on the site (for example, Site1). This is because the script calculates the highest port number in the `mongoConfig` on the site where you are creating shards. This creates clash between the replica-sets on both sites. Since the port number which it allocates might overlap with the port number of `mongoConfig` on other site (for example, Site2). This is the reason why there should be some gap in the port numbers allocated between both the sites.

Step 1 Log in to Cluster Manager as a root user.

Step 2 Modify the `/etc/broadhop/gr_cluster.conf` file. For example, if `/etc/broadhop/qns.conf` file has the following entries for Site ID:.

```
-DSiteId=clusterA

-DRemoteSiteId=clusterB
```

a) Add cluster name that is, clusterA followed by pcrfclient01/pcrfclient02 management interface public IP address of the Primary ClusterA.

For example, `clusterA:`*`a.b.c.d`*`,`*`w.x.y.z`*

where,

a.b.c.d is the pcrfclient01 management interface public IP address of the Primary ClusterA.

a.b.c.d is the pcrfclient02 management interface public IP address of the Primary ClusterA.

b) On next line add remote site name that is, clusterB followed by pcrfclient01/pcrfclient02 Management-interface public IP address of the Secondary ClusterB (these public IP addresses should be pingable from Site1).

For example, `clusterA:`*`e.f.g.h`*`,`*`p.q.r.s`*

where,

e.f.g.h is the pcrfclient01 management interface public IP address of the Secondary ClusterB.

p.q.r.s is the pcrfclient02 management interface public IP address of the Secondary ClusterB.

These entries need to match with site name entries given in `qns.conf` file.

File contents look like:

```
cat /etc/broadhop/gr_cluster.conf
#<site name>:<pcrfclient01 IP address>:<pcrfclient02 IP address>
#Primary sites
clusterA:a.b.c.d,w.x.y.z
#Secondary sites
clusterB:e.f.g.h,p.q.r.s
```

Step 3 **Verify MongoConfig**: Do not miss to add `#SITE`*`x`*`_START` and `#SITE`*`x`*`_END` tags to the block of replica set entries in `/etc/broadhop/mongoConfig.cfg` file, where *x* is the site number. To add these tags at proper location, refer to sample configuration file (`geo_mongoconfig_template`) present in `/etc/broadhop` directory. The SiteIDs must be obtained from `/etc/broadhop/qns.conf` file from the field `-DSiteID`.

Example:

For example, if Site1 (clusterA_PRI) and Site2 (clusterA_SBY) are the two sites, then you need to add Site1 and Site2 entries in `mongoconfig.cfg` file as per the sample configuration file (`geo_mongoconfig_template`) present in `/etc/broadhop` directory.

Example:

To monitor the Arbiter VM alarms, the `mongoConfig.cfg` file must identify the local and remote replica-sets separately. The sets should be marked by SiteID of the site. This SiteID must be obtained from `/etc/broadhop/qns.conf` file from the field `-DSiteID`. For marking/separating the two sites replica-sets, identify the start and end of the replica-sets.

Note Make sure case sensitivity for SiteID as well as START and END tags.

For example, one site is having SiteId as Site1 (clusterA_PRI) and other has Site2 (clusterA_SBY). So, the `mongoconfig.cfg` looks like:

```
#Site1_START
#clusterA_PRI_START
.
.
<All replica sets for Site1.>
.
.
#clusterA_PRI_END
#Site1_END

#Site2_START
#clusterA_SBY_START
.
.
<All Replica sets for Site2.>
.
.
#clusterA_SBY_END
#Site2_END
```

Step 4 To install the database and synchronize the `mongoConfig.cfg` file across the cluster execute the following commands:

```
/var/qps/install/current/scripts/build/build_etc.sh

/var/qps/bin/update/syncconfig.sh
```

Step 5 Set priority using `set_priority.sh` command. The following are example commands:

```
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db session
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db spr
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db admin
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db balance
```

The primary member of individual replica-sets are on respective sites.

Step 6 Verify replica set status and priority is set correctly using the following command:

```
diagnostics.sh --get_replica_status
```

Step 7 When the Session replication is configured then the Host collection of the Cluster database should have all the “admin replica-set” members, entries with Internal and Replication VLAN IP's.

By default, `db.hosts` file gets populated if you configure `/etc/broadhop/gr_cluster.conf` file. If the entries are not present then use the following commands to add these entries (XX is internal/replication IP address of “admin replica-set” and YY is siteId (defined in `qns.conf`):

```
mongo --host <admin DB primary host> --port <admin DB port> clusters
> db.hosts.update({"ip" : "XX"},{"siteName" : "YY", "ip" : "XX"}, true)
```

Example:

```
mongo CA-PRI-sessionmgr02:27721/clusters
MongoDB shell version: 2.6.3
connecting to: CA-PRI-sessionmgr02:27721/clusters
set05:PRIMARY> db.hosts.find()
{ "_id" : ObjectId("545e0596f4ce7b3cc119027d"), "siteName" : "clusterA_PRI", "ip" :
"192.168.109.127" }
{ "_id" : ObjectId("545e0596f4ce7b3cc119027e"), "siteName" : "clusterA_PRI", "ip" :
"192.168.109.128" }
{ "_id" : ObjectId("545e0596f4ce7b3cc1190281"), "siteName" : "clusterA_SBY", "ip" :
"192.168.109.227" }
{ "_id" : ObjectId("545e0596f4ce7b3cc1190282"), "siteName" : "clusterA_SBY", "ip" :
"192.168.109.228" }
{ "_id" : ObjectId("545e0596f4ce7b3cc119027d"), "siteName" : "clusterA_PRI", "ip" : "11.11.11.127"
```

```
}
{ "_id" : ObjectId("545e0596f4ce7b3cc119027e"), "siteName" : "clusterA_PRI", "ip" : "11.11.11.128"
}
{ "_id" : ObjectId("545e0596f4ce7b3cc1190281"), "siteName" : "clusterA_SBY", "ip" : "11.11.11.227"
}
{ "_id" : ObjectId("545e0596f4ce7b3cc1190282"), "siteName" : "clusterA_SBY", "ip" : "11.11.11.228"
}
```

1. (Optional) By default, db.hosts gets populated if there is a difference between IP addresses of sessionmgr* VMs in `/etc/hosts` file on both sites.

Example:

For sessionmgr01 SITE-A, in `/etc/hosts` file, if the entry is: `10.10.10.1 sessionmgr01 sessionmgr01-SITE-A`

and for SITE-B on sessionmgr01, in `/etc/hosts`, if the entry is: `172.20.20.1 sessionmgr01-SITE-A`

As, IP addresses of sessionmgr VMs are different in this case, user needs to run the following scripts on both SITEs.

```
cd /var/qps/bin/support/mongo/; ./set_clusterinfo_in_admindb.sh
```

Step 8 From this Cluster Manager, copy `mongoConfig.cfg` and `gr_cluster.conf` files to peer Cluster Managers (CM).

Balance Backup Database Configuration

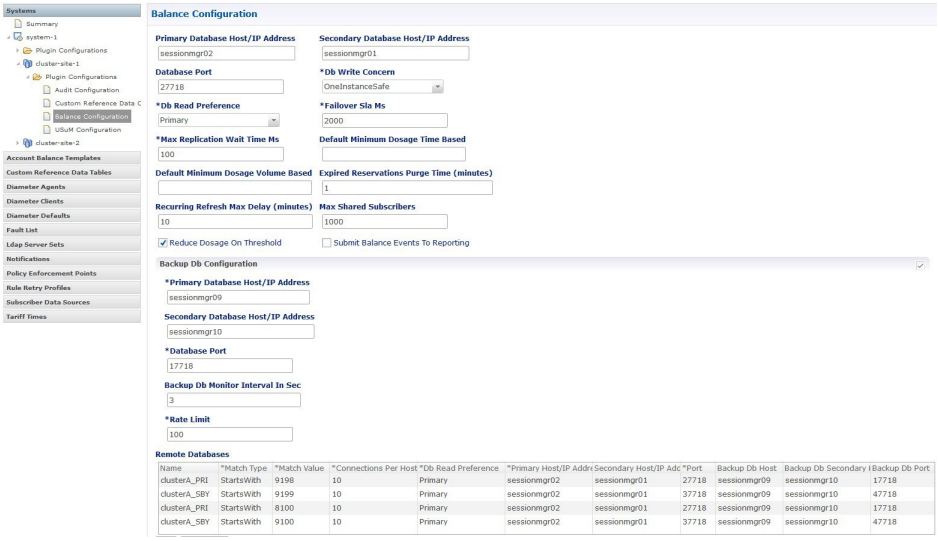
CPS provides extra high availability for balance database during failover. During failover or switchover, or when primary is not available due to network reachability, balance writes happen in the backup database. After primary database is available, the records in backup database are reconciled with the primary.

The balance backup database configuration in mongoConfig appears like any other balance database with two members of replica-set on a given site. The replica-set must be created in the same manner in which regular balance database replica-sets are created.

Step 1 In Policy Builder, click **Reference Data** > **Systems** > ***name of your primary system*** > **Plugin Configurations** and select **Balance Configuration** from right side. In **Balance Configuration**, configure the primary database information. For parameter description, refer to *CPS Mobile Configuration Guide*.

An example configuration is shown:

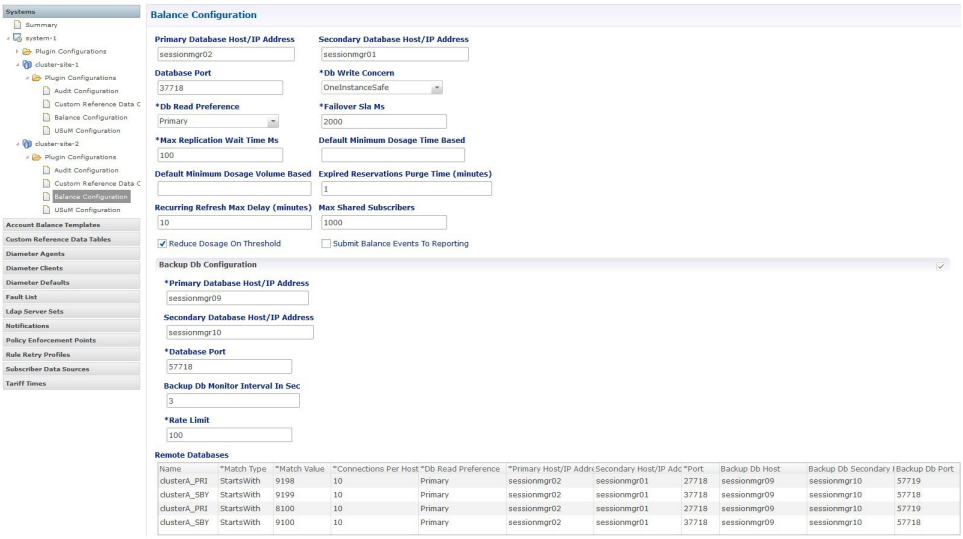
Figure 14: Balance Backup Database Configuration - 1



Step 2 In Policy Builder, click **Reference Data** > **Systems** > ***name of your backup system*** > **Plugin Configurations** and select **Balance Configuration** from right side. In **Balance Configuration**, configure the backup database information. For parameter description, refer to *CPS Mobile Configuration Guide*.

An example configuration is shown:

Figure 15: Balance Backup Database Configuration - 2



Step 3 The following is an example output for balance backup database:

```
diagnostics.sh --get_re
```

The balance database replica-sets for Site1 and Site2 are displayed in the example output.

```
CPS Diagnostics GR Multi-Node Environment
--------------------------
Checking replica sets...
|-----------------------------------------------------------------------------------------------------------------------------------------|
```

```
| Mongo:x.x.x                      MONGODB REPLICA-SETS STATUS INFORMATION OF SITE1           Date
: 2016-09-26 10:59:52 |
|-------------------------------------------------------------------------------------------------------------------------|
|  SET NAME  -  PORT  : IP ADDRESS   - REPLICA STATE -           HOST NAME         -  HEALTH  -
LAST SYNC   - PRIORITY |
|-------------------------------------------------------------------------------------------------------------------------|
| ADMIN:set08
                     |
|  Member-1  -  27721 : 172.20.18.54  -  ARBITER      -  L2-CA-ARB-sessionmgr15    - ON-LINE  -
--------    -    0     |
|  Member-2  -  27721 : 172.20.17.83  -  PRIMARY      -  L2-CA-PRI-sessionmgr09    - ON-LINE  -
--------    -    4     |
|  Member-3  -  27721 : 172.20.17.87  -  SECONDARY    -  L2-CA-PRI-sessionmgr10    - ON-LINE  -
1 sec       -    3     |
|  Member-4  -  27721 : 172.20.19.53  -  SECONDARY    -  L2-CA-SEC-sessionmgr09    - ON-LINE  -
1 sec       -    2     |
|  Member-5  -  27721 : 172.20.19.57  -  SECONDARY    -  L2-CA-SEC-sessionmgr10    - ON-LINE  -
1 sec       -    1     |
|-------------------------------------------------------------------------------------------------------------------------|
| BALANCE:set05
                     |
|  Member-1  -  27718 : 172.20.18.54  -  ARBITER      -  L2-CA-ARB-sessionmgr15    - ON-LINE  -
--------    -    0     |
|  Member-2  -  27718 : 172.20.17.40  -  PRIMARY      -  L2-CA-PRI-sessionmgr02    - ON-LINE  -
--------    -    4     |
|  Member-3  -  27718 : 172.20.17.38  -  SECONDARY    -  L2-CA-PRI-sessionmgr01    - ON-LINE  -
0 sec       -    3     |
|  Member-4  -  27718 : 172.20.19.29  -  SECONDARY    -  L2-CA-SEC-sessionmgr02    - ON-LINE  -
0 sec       -    2     |
|  Member-5  -  27718 : 172.20.19.27  -  SECONDARY    -  L2-CA-SEC-sessionmgr01    - ON-LINE  -
0 sec       -    1     |
|-------------------------------------------------------------------------------------------------------------------------|
| BALANCE:set09a
                     |
|  Member-1  -  47718 : 172.20.18.54  -  ARBITER      -  L2-CA-ARB-sessionmgr15    - ON-LINE  -
--------    -    0     |
|  Member-2  -  47718 : 172.20.17.87  -  PRIMARY      -  L2-CA-PRI-sessionmgr10    - ON-LINE  -
--------    -    2     |
|  Member-3  -  47718 : 172.20.17.83  -  SECONDARY    -  L2-CA-PRI-sessionmgr09    - ON-LINE  -
0 sec       -    1     |
|-------------------------------------------------------------------------------------------------------------------------|
| BALANCE:set10a
                     |
|  Member-1  -  17718 : 172.20.18.54  -  ARBITER      -  L2-CA-ARB-sessionmgr15    - ON-LINE  -
--------    -    0     |
|  Member-2  -  17718 : 172.20.17.83  -  PRIMARY      -  L2-CA-PRI-sessionmgr09    - ON-LINE  -
--------    -    2     |
|  Member-3  -  17718 : 172.20.17.87  -  SECONDARY    -  L2-CA-PRI-sessionmgr10    - ON-LINE  -
0 sec       -    1     |
|-------------------------------------------------------------------------------------------------------------------------|
| REPORTING:set07
                     |
|  Member-1  -  27719 : 172.20.18.54  -  ARBITER      -  L2-CA-ARB-sessionmgr15    - ON-LINE  -
--------    -    0     |
|  Member-2  -  27719 : 172.20.17.44  -  PRIMARY      -  L2-CA-PRI-sessionmgr04    - ON-LINE  -
--------    -    2     |
|  Member-3  -  27719 : 172.20.17.42  -  SECONDARY    -  L2-CA-PRI-sessionmgr03    - ON-LINE  -
0 sec       -    1     |
|-------------------------------------------------------------------------------------------------------------------------|
| SESSION:set01
                     |
|  Member-1  -  27717 : 172.20.18.54  -  ARBITER      -  L2-CA-ARB-sessionmgr15    - ON-LINE  -
--------    -    0     |
|  Member-2  -  27717 : 172.20.17.38  -  PRIMARY      -  L2-CA-PRI-sessionmgr01    - ON-LINE  -
```

```
--------    -    4     |
|  Member-3  -  27717 : 172.20.17.40  -  SECONDARY    -  L2-CA-PRI-sessionmgr02    - ON-LINE  -
0 sec      -    3     |
|  Member-4  -  27717 : 172.20.19.27  -  SECONDARY    -  L2-CA-SEC-sessionmgr01    - ON-LINE  -
0 sec      -    2     |
|  Member-5  -  27717 : 172.20.19.29  -  SECONDARY    -  L2-CA-SEC-sessionmgr02    - ON-LINE  -
0 sec      -    1     |
|-------------------------------------------------------------------------------------------------------------------|
|-------------------------------------------------------------------------------------------------------------------|
| Mongo:x.x.x                        GODB REPLICA-SETS STATUS INFORMATION OF SITE2             Date
: 2016-09-26 11:00:06 |
|-------------------------------------------------------------------------------------------------------------------|
|  SET NAME  -  PORT  : IP ADDRESS       - REPLICA STATE -           HOST NAME      -  HEALTH  -
LAST SYNC   - PRIORITY |
|-------------------------------------------------------------------------------------------------------------------|
| BALANCE:set25
                    |
|  Member-1  -  37718 : 172.20.18.54     -  ARBITER      -  L2-CA-ARB-sessionmgr15  - ON-LINE  -
--------    -     0     |
|  Member-2  -  37718 : 172.20.19.29     -  PRIMARY      -  L2-CA-SEC-sessionmgr02  - ON-LINE  -
--------    -     4     |
|  Member-3  -  37718 : 172.20.19.27     -  SECONDARY    -  L2-CA-SEC-sessionmgr01  - ON-LINE  -  0
 sec      -     3     |
|  Member-4  -  37718 : 172.20.17.38     -  SECONDARY    -  L2-CA-PRI-sessionmgr01  - ON-LINE  -  0
 sec      -     2     |
|  Member-5  -  37718 : 172.20.17.40     -  SECONDARY    -  L2-CA-PRI-sessionmgr02  - ON-LINE  -  0
 sec      -     1     |
|-------------------------------------------------------------------------------------------------------------------|
| BALANCE:set26
                    |
|  Member-1  -  57719 : 172.20.18.54     -  ARBITER      -  L2-CA-ARB-sessionmgr15  - ON-LINE  -
--------    -     0     |
|  Member-2  -  57719 : 172.20.19.57     -  PRIMARY      -  L2-CA-SEC-sessionmgr10  - ON-LINE  -
--------    -     2     |
|  Member-3  -  57719 : 172.20.19.53     -  SECONDARY    -  L2-CA-SEC-sessionmgr09  - ON-LINE  -  0
 sec      -     1     |
|-------------------------------------------------------------------------------------------------------------------|
| BALANCE:set27
                    |
|  Member-1  -  57718 : 172.20.18.54     -  ARBITER      -  L2-CA-ARB-sessionmgr15  - ON-LINE  -
--------    -     0     |
|  Member-2  -  57718 : 172.20.19.53     -  PRIMARY      -  L2-CA-SEC-sessionmgr09  - ON-LINE  -
--------    -     2     |
|  Member-3  -  57718 : 172.20.19.57     -  SECONDARY    -  L2-CA-SEC-sessionmgr10  - ON-LINE  -  0
 sec      -     1     |
|-------------------------------------------------------------------------------------------------------------------|
```

Session Cache Hot Standby

Important Cisco recommends to configure standby session for GR.

CPS runs a distributed database called MongoDB. MongoDB uses a replication concept for high availability called replica-sets. A replica-set is made up of independent MongoDB instances that run in one of the following three modes:

- Primary: A primary database is the only database available that can accept writes.
- Secondary: A secondary database is a database that is read only and is actively synchronizing to a primary database by replaying the primary's oplog (operations log) on the local node.
- Recovering: A secondary database that is currently synchronizing to the primary and has not caught up to the primary.

Session data is highly concurrent, the application always reads and writes from the primary database. The secondary database(s) provide HA for the primary in the event of VM shutdown or process shutdown. Hot standby session cache replica set is configured to take over the load while primary database is failing over to secondary session cache database. In this fail-over process, it minimizes the call failures and provides high system availability.

Prerequisites

- Hot standby replica-set must be created on different blades (for maximum protection).
- Admin database and session cache databases must be separate replica-sets.
- Hot standby replica-set should be added to shard configuration as backup database true.

Configuration

Step 1 The hotstandby database must be configured just like any other session cache database in mongo config and a replica-set needs to be created.

The following is an example backup database configuraton in mongoDB:

```
[SESSION-SET1] SETNAME=set01
ARBITER=pcrfclient01-prim-site-1:37718
ARBITER_DATA_PATH=/data/sessions.3
MEMBER1=sessionmgr01-site1:27718
MEMBER2=sessionmgr02-site1:27718
DATA_PATH=/data/sessions.3
[SESSION-SET1-END]
```

This needs to be created using `build_set` script for VMware. For more information, refer to *CPS Installation Guide for VMware*.

For OpenStack, user `/api/system/config/replica-sets`. For more information, refer to *CPS Installation Guide for OpenStack*.

Step 2 Verify CPS application is running on both the sites (pcrfclient01 and pcrfclient02) and without any application errors.

Example:

By executing `diagnostics.sh` script you can get the diagnostics of application. The `diagnostics.sh` output should not contain any application errors.

Step 3 Verify whether shard command is available in OSGi console or not. From pcrfclient01, login as root user into the OSGi console and run the help.
You can find the following shard command:

```
telnet qns01 9091
osgi> help
---QNS Commands---
```

```
     reload - Reload reference data
     genpassword <db password>
---Sharing Commands---
addshard seed1[,seed2] port db-index [backup]
rebalance
migrate
---Controlling the Console---
more - More prompt for console output
disconnect - Disconnects from telnet session
help <commmand> - Display help for the specified command.
```

Step 4 To configure hot standby session management, execute the following commands:

```
telnet qns01 9091
addshard sessionmgr03,sessionmgr04 27717 1 Site1 backup
addshard sessionmgr03,sessionmgr04 27717 2 Site1 backup
addshard sessionmgr03,sessionmgr04 27717 3 Site1 backup
addshard sessionmgr03,sessionmgr04 27717 4 Site1 backup
rebalance
migrate
disconnect
y
```

Step 5 To verify the configuration:

1. Login to primary administration database using port#<admin DB port> and verify the collection shards in shading database.

```
mongo sessionmgr01:27721
MongoDB shell version: 2.6.3
connecting to: sessionmgr01:27721/test
set05:PRIMARY> use sharding
switched to db sharding
set05:PRIMARY> db.shards.find()
{ "_id" : 1, "seed_1" : "sessionmgr03", "seed_2" : "sessionmgr04", "port" : 27717, "db" :
"session_cache_2", "online" :
true,
"count" : NumberLong(0), "backup_db" : true }
{ "_id" : 2, "seed_1" : "sessionmgr03", "seed_2" : "sessionmgr04", "port" : 27717, "db" :
"session_cache_3", "online" :
true,
"count" : NumberLong(0), "backup_db" : true }
{ "_id" : 3, "seed_1" : "sessionmgr04", "seed_2" : "sessionmgr04", "port" : 27717, "db" :
"session_cache_4", "online" :
true,
"count" : NumberLong(0), "backup_db" : true }
{ "_id" : 4, "seed_1" : "sessionmgr03", "seed_2" : "sessionmgr04", "port" : 27717, "db" :
"session_cache", "online" :
true,
"count" : NumberLong(0), "backup_db" : true }
set05:PRIMARY
```

Failover Detection

There are three possible ways a MongoDB node can fail and trigger a fail over to another node:

1. Replica set step down: This scenario is the cleanest method since it disconnects all client sockets and immediately initiates a new election for a master node.

2. Process abort: This scenario occurs when a node aborts due to an internal failure. Since this is unplanned, the other replica nodes will not request a new election for a master node until a majority of the nodes have detected the master as down.
3. VM power off: This scenario occurs when a node is powered off without a proper shutdown. In this case sockets are usually hung on all clients and the other replica nodes will not request a new election for a master node until a majority of the nodes have detected the master as down.

The Cisco Policy Server detects client failure by:

1. Utilizing the pre-packaged MongoDB Java driver software to detect failure.
2. Detecting server power off via server pings to rapidly detect power off situations.

Limitation

You can configure only one backup database. Thus, in GR Active/Standby configuration, if you configure backup database on the standby site, during local primary to local secondary database fail over on active site, the sessions would be saved on the backup database which is on secondary site. This might increase cross-site traffic temporarily.

Policy Builder Configuration

Step 1 Configure and publish Policy Builder changes from each site. Use **about.sh** command to find out Policy Builder URL.

Cisco recommends to configure and publish Policy Builder data separately from each site. But if the user wants to publish Policy Builder data from single site to all other sites then it is difficult to access Policy Builder data from other sites when the primary site goes down.

To access Policy Builder data from other sites when primary site is down, refer Access Policy Builder from Standby Site when Primary Site is Down, on page 46.

Step 2 Set appropriate Primary Database IP address, Secondary Database IP address and Port numbers for the following plug-ins:

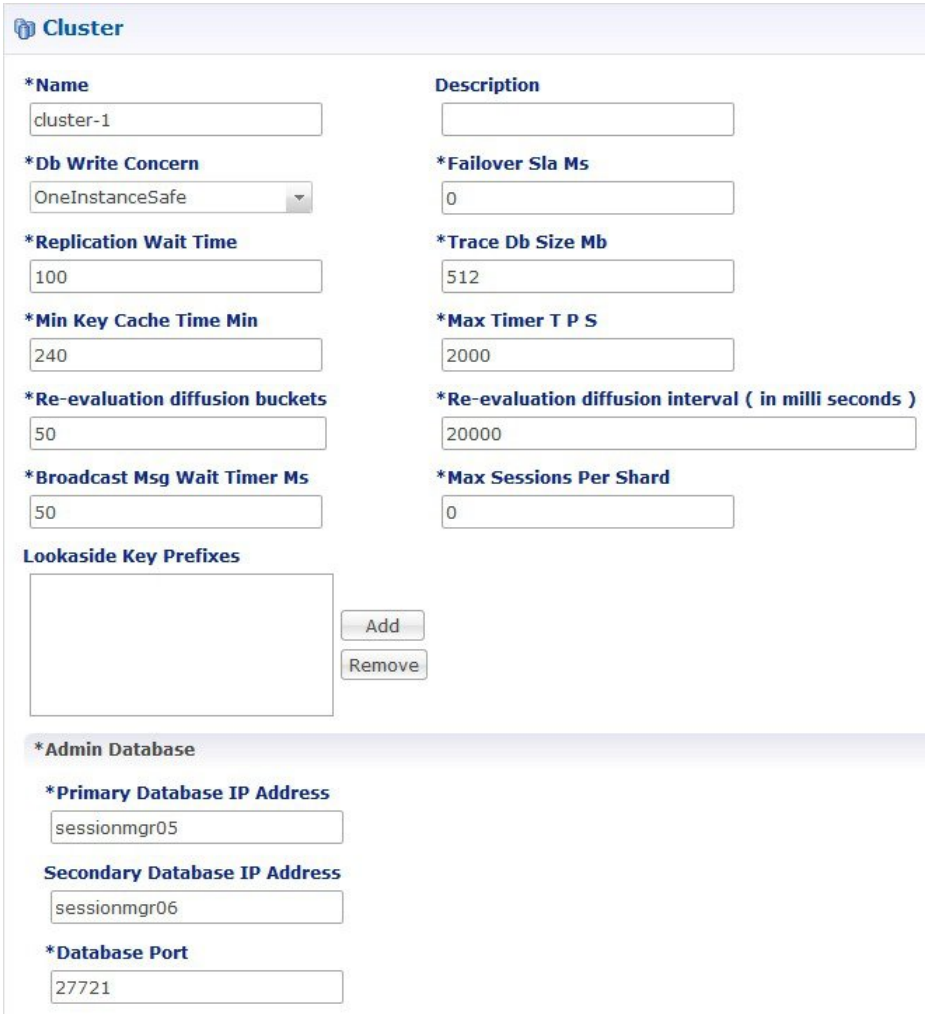
- USuM Configuration
- Balance Configuration
- Custom Reference Data Configuration
- Voucher Configuration
- Audit Configuration

Step 3 Set **Balance Configuration** > **Db Read Preference** as **SecondaryPreferred** for all databases except balance database.

Figure 16: Db Read Preference

An example **Cluster** configuration is given:

Figure 17: Policy Builder Screen



Also update **Lookaside Key Prefixes** and **Admin Database** sections. For more information, refer to *CPS Mobile Configuration Guide*.

Step 4 It is recommended to publish Policy Builder changes from each site. If a user is using primary site to publish Policy Builder changes then publishing into all the following cluster repositories is not recommended:

Table 10: Publishing

Cluster	Publish URL
CA-PRI	http://<*Management interface public IP address of CA-PRI-pcrfclient01*>/repos/run
CA-SEC	http://< *Management interface public IP address of CA-SEC-pcrfclient01*>/repos/run

Step 5 Add all the above repositories. Repository and Publish screen looks like:

Figure 18: Repository Screen

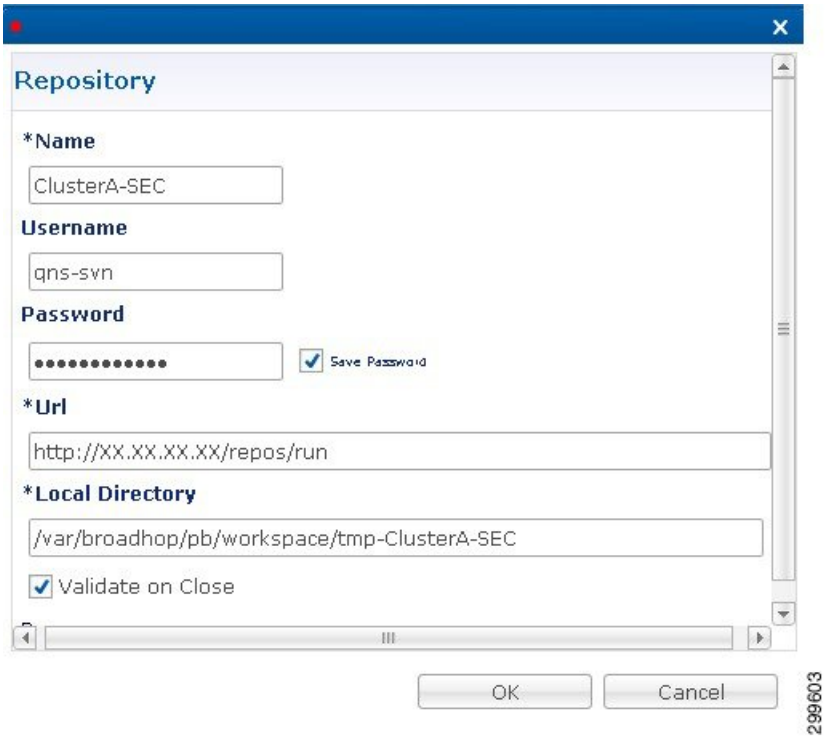
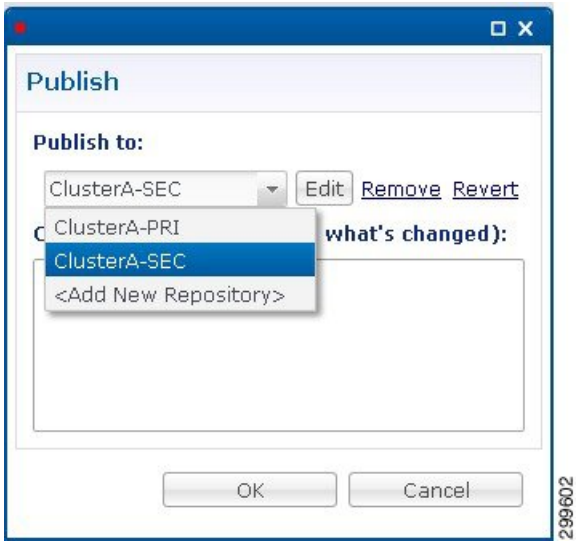


Figure 19: Publish Screen



Step 6 Validate both setups using **diagnostics.sh**, after publishing the repository (wait for five minutes) OR follow the Validate VM Deployment section in the *Cisco Policy Suite Installation Guide* for this release.

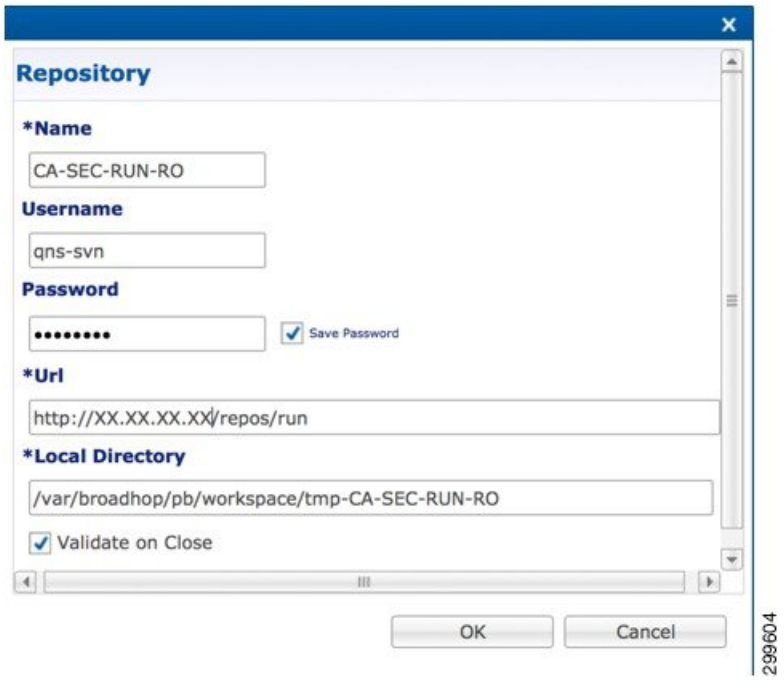
Access Policy Builder from Standby Site when Primary Site is Down

Step 1 This is recommended only when primary site is down and secondary is only for reading/viewing purpose. It is applicable only where user publishes policy builder data from single site that is, primary site to all other sites.

Step 2 Open Policy Builder from secondary site (use **about.sh** command to find out PB URL).

Step 3 Create new data repository SEC-RUN-RO using URL 'http://< Management interface public IP address of secondary pcrfclient01>/repos/run', screen looks like:

Figure 20: New Data Repository



Step 4 Access Policy Builder from secondary site using newly created repository.

qns.conf Configuration Changes for Session Replication

The following changes are required in `qns.conf` file when session replication is required for active/active or active/standby GR deployments.

For active/active GR deployment, Geo HA feature needs to be enabled. For more information, refer to Active/Active Geo HA - Multi-Session Cache Port Support, on page 129.

Step 1 Add the following GR related parameters in `/etc/broadhop/qns.conf` file of Cluster A Primary cluster manager VM that is, CA-PRI-cm:

```
-DGeoSiteName=clusterA_PRI
-DSiteId=clusterA_PRI
-DRemoteSiteId=clusterA_SBY

-DheartBeatMonitorThreadSleepMS=500
-Dcom.mongodb.updaterConnectTimeoutMS=1000
-Dcom.mongodb.updaterSocketTimeoutMS=1000
-DdbConnectTimeout=1200
-Dmongo.client.thread.maxWaitTime=1200
-DdbSocketTimeout=600
-DclusterFailureDetectionMS=2000
```

Step 2 Add the following GR related parameters in `/etc/broadhop/qns.conf` file of Cluster A Secondary cluster manager VM that is, CA-SEC-cm:

```
-DGeoSiteName=clusterA_SBY
-DSiteId=clusterA_SBY
-DRemoteSiteId=clusterA_PRI

-DheartBeatMonitorThreadSleepMS=500
-Dcom.mongodb.updaterConnectTimeoutMS=1000
-Dcom.mongodb.updaterSocketTimeoutMS=1000
-DdbConnectTimeout=1200
-Dmongo.client.thread.maxWaitTime=1200
-DdbSocketTimeout=600
-DclusterFailureDetectionMS=2000
```

Step 3 For multi-cluster, the following setting should be present only for GR (multi-cluster) CPS deployments:

```
-DclusterFailureDetectionMS=1000
```

Note In an HA or GR deployment with local chassis redundancy, the following setting should be set to true. By default, this is set to false.

```
-Dremote.locking.off
```

Step 4 Create `etc` directory on each cluster using `/var/qps/install/current/scripts/build/build_etc.sh` script.

Step 5 Copy the changes in `qns.conf` to other VMs:

```
copytoall.sh /etc/broadhop/qns.conf /etc/broadhop/qns.conf
```

Step 6 Restart all software components on the target VMs:

```
restartall.sh
```

Step 7 Validate setup using `diagnostics.sh` or follow *Validate VM Deployment* section in the *CPS Installation Guide for VMware* for this release.

Configurations to Handle Database Failover when Switching Traffic to Standby Site Due to Load Balancer Fail/Down

Note To understand traffic switch over, refer to Load Balancer VIP Outage, on page 148.

Step 1 Add the list of databases that needs to be migrated to primary on other site after traffic switch over in `mon_db_for_lb_failover.conf` file (`/etc/broadhop/mon_db_for_lb_failover.conf`) in Cluster Manager.

Note Contact your Cisco Technical Representative for more details.

Add the following content in the configuration file (`mon_db_for_lb_failover.conf`):

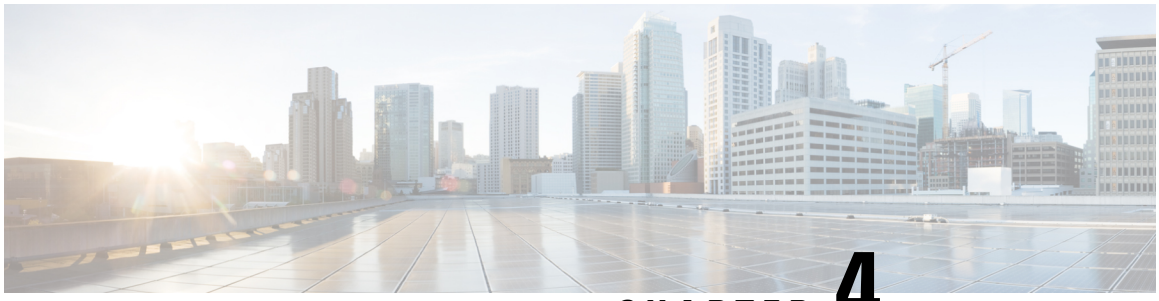
The following is an example and needs to be changed based on your requirement.

```
#this file contains set names that are available in mongoConfig.cfg. Add set names one below other.
#Refer to README in the scripts folder.
SESSION-SET1
SESSION-SET2
```

```
BALANCE-SET1
SPR-SET1
```

Step 2 Rebuild etc directory on cluster by executing the following command:

```
/var/qps/install/current/scripts/build/build_etc.sh
```



CHAPTER 4

GR Installation - OpenStack

- GR Installation - OpenStack, on page 51
- Arbiter Installation on OpenStack, on page 55
- Configuration Parameters - GR System, on page 58

GR Installation - OpenStack

The examples given in the steps is for your reference only. You need to modify them based on your GR deployments.

Important Copying YAML and environment files from this document is not recommended. The files are provided for your reference only.

Before you begin

- Download the latest ISO build.
- Create CPS VMs using Heat template or Nova boot commands on all GR sites. In the following section, heat template has been considered as an example to deploy GR (here examples are site1, site2 and arbiter) sites.

For more information, refer to *CPS Installation Guide for OpenStack*

Step 1 Create instances for site1, site2 and Arbiter. Wait till they are cluman ready.

Check the readiness status of the Cluster Manager VM on all the sites using the API: `GET http://<Cluster Manager IP>:8458/api/system/status/cluman`.

External replication VLAN information should be added for each VM in the `hot-cps.env` and `hot-cps.yaml` for communication between GR sites.

Refer to Sample Heat Environment File, on page 151 and Sample Heat Template File, on page 153 for sample configuration of site1. For site2 similar files need to be created by modifying hostname, IP addresses and so on.

For Arbiter, refer to Arbiter Installation on OpenStack, on page 55.

Step 2 Load CPS configuration files on each site: Refer to */api/system/config/* section in *CPS Installation Guide for OpenStack*.

GR Installation - OpenStack

In `CPS_system_config.yaml` file, give consideration to the following mentioned items:

- Under Additional Host section, add session manager information of other site (site1 or site2) and arbiter.
- Mongo replica members should include the site identifier to differentiate database host such as, sessionmgr01-site1 from sessionmgr01-site2. Database host names (such as sessionmgr01-site1) needs to be modified according to the your GR deployment in template file.
- Update `policyServerConfig:` section according to your GR deployment.
- Internal/management/external IPs need to modified in `hosts:` and `additionalhosts:` section according to your GR deployment.
- In `additionalhosts:` section, other site session manager host entry should be added with alias `psessionmgr`*xx*.

For sample configurations, refer to Sample YAML Configuration File - site1, on page 178 and Sample YAML Configuration File - site2, on page 186.

Note If you want to add the MongoDB authentication, refer to *Configuration Parameters - HA System* section in *CPS Installation Guide for OpenStack*. You need to mention password for all the sites separately using API and that must be same for all the sites.

If you want to enable mongo authentication you need to add the following parameters under `config:` section in YAML configuration file:

```
config:
  dbAuthenticationEnabled: "true"
  dbAuthenticationAdminPasswd: "XXXX"
  dbAuthenticationReadonlyPasswd: "YYYY"
  dbAuthenticationEncryption: "false"
```

where, *XXXX* and *YYYY* are encrypted passwords.

Step 3 (Optional) To confirm the configuration was loaded properly onto the Cluster Manager VM on each site, perform a GET with the API:

```
GET http://<Cluster Manager IP>:8458/api/system/config/
```

Step 4 Apply the configuration using the following API on each site:

```
POST http://<Cluster Manager IP>:8458/api/system/config/apply
```

Refer to *Apply the Loaded Configuration* section in *CPS Installation Guide for OpenStack* for more information.

This API applies the CPS configuration file, triggers the Cluster Manager VM to deploy and bring up all CPS VMs on each site, and performs all post-installation steps.

Important Wait for approx 15 minutes for the API to complete the all post-installation steps.

Step 5 In your mongo YAML file, add other site members as secondary-members and local site members as primary members for respective databases depending on your GR deployment.

For sample configuration, refer to Sample Mongo Configuration File - site1, on page 193 and Sample Mongo Configuration File - site2, on page 195.

Step 6 After updating the mongo YAML files, apply them using the */api/system/mongo/config* API on each site with their YAML file.

Refer to */api/system/mongo/config* section in *CPS Installation Guide for OpenStack*.

Note This step will not create replica-set for added members. It will create only new mongo configuration file on each site.

Step 7 Add remote site pcrfclient IPs in respective `gr_cluster.yaml` files.

For sample configuration, refer to Sample GR Cluster Configuration File - site1, on page 199 and Sample GR Cluster Configuration File - site2, on page 199.

Step 8 Execute below APIs from respective sites to update the GR cluster information and populate respective ADMIN host database.

For example:

```
curl -i -X PATCH http://installer-site1:8458/api/system/config/application-config -H "Content-Type:
application/yaml" --data-binary @gr_cluster.yaml
```

```
curl -i -X PATCH http://installer-site2:8458/api/system/config/application-config -H "Content-Type:
application/yaml" --data-binary @gr_cluster2.yaml
```

For sample configuration, refer to Sample GR Cluster Configuration File - site1, on page 199 and Sample GR Cluster Configuration File - site2, on page 199.

Verify whether:

- Remote pcrfclient IPs are populated correctly in `/etc/broadhop/gr_cluster.conf` file.
- ADMIN database has been populated correctly, run `mongo sessionmgr01-site1:27721/clusters --eval "db.hosts.find()"` and `mongo sessionmgr01-site2:27769/clusters --eval "db.hosts.find()"` on primary database member on site-1 and site-2 console.

Step 9 If all sites are deployed and configured, then create geo replica-sets between site1 and site2:

a) Combine both sites mongo YAML files to be used in your GR deployment.

For sample configuration, refer to Sample Mongo GR Configuration File, on page 196.

b) After combining YAML files, post the combined file on both sites except arbiter. For more information, refer to */api/system/mongo/config* section in *CPS Installation Guide for OpenStack*.

For example:

```
curl -i -X PUT http://installer-site1:8458/api/system/mongo/config -H "Content-Type:
application/yaml" --data-binary @mongogr.yaml
```

```
curl -i -X PUT http://installer-site2:8458/api/system/mongo/config -H "Content-Type:
application/yaml" --data-binary @mongogr.yaml
```

c) Remove unused replica-sets from site2 using `/var/qps/bin/support/mongo/build_set.sh` script.

In the sample configuration file common SPR, Balance and ADMIN between site1 and site2 are being used, thus these replica-sets can be removed from site2.

d) Add members to replica-set. This API needs to be executed from primary site (site1) only.

For example:

```
curl -i -X POST http://installer-site1:8458/api/system/mongo/action/addMembers
```

e) Configure the priority using the following APIs:

```
curl -i -X PATCH http://installer-site1:8458/api/system/config/replica-sets -H "Content-Type:
application/yaml" --data-binary @setPriority-site1.yaml
```

GR Installation - OpenStack

```
curl -i -X PATCH http://installer-site1:8458/api/system/config/replica-sets -H "Content-Type:
application/yaml" --data-binary @setPriority-site2.yaml
```

For sample configuration, refer to Sample Set Priority File - site1, on page 199 and Sample Set Priority File - site2, on page 199.

Step 10 Create appropriate clusters in Policy Builder such as, 'Cluster-SITE1' for site1 and 'Cluster-SITE2' for site2 and update Primary Database IP Address, Secondary Database IP Address and Database port number based on mongo configuration and publish to the respective sites depending on your GR deployment.

For more information, refer to Policy Builder Configuration, on page 43.

Step 11 Run `diagnostics.sh` on both sites to display the current state of the system. Make sure there are no error on both the sites.

Step 12 Modify/add shard on respective sites. It contains each site session replication sets with backup database.

For example:

```
curl -i -X PATCH http://installer-site1:8458/api/system/config/replica-sets/ -H "Content-Type:
application/yaml" --data-binary @modify_shard.yaml
```

```
curl -i -X PATCH http://installer-site2:8458/api/system/config/replica-sets/ -H "Content-Type:
application/yaml" --data-binary @modify_shard2.yaml
```

For sample configuration, refer to Sample Shard Configuration File - site1, on page 200 and Sample Shard Configuration File - site2, on page 200.

Step 13 Modify/add ring: It contains only session replica-sets and not backup database. This API needs to be executed from primary site.

For example:

```
curl -i -X PATCH http://installer-site1:8458/api/system/config/replica-sets/ -H "Content-Type:
application/yaml" --data-binary @modify_ring.yaml
```

For sample configuration, refer to Sample Ring Configuration File, on page 200.

Step 14 Add geo-site lookup for both sites.

For example:

```
curl -i -X PATCH http://installer-site1:8458/api/system/config/application-config -H "Content-Type:
application/yaml" --data-binary @geositelookup.yaml
```

```
curl -i -X PATCH http://installer-site2:8458/api/system/config/application-config -H "Content-Type:
application/yaml" --data-binary @geositelookup2.yaml
```

For sample configuration, refer to Sample Geo Site Lookup Configuration File - site1, on page 200 and Sample Geo Site Lookup Configuration File - site2, on page 201.

Note The pattern matching is supported for site lookup mapping. In case the incoming host/realm does not match any of the values configured under LookupValues, request is dropped with the following exception in log:

```
GeoHASiteMappingNotFound - No realm/host to site mapping matched for:
<incoming value>
```

Step 15 Add geo tags in replica-sets for both sites.

For example:

```
curl -i -X PATCH http://installer-site1:8458/api/system/config/replica-sets/ -H "Content-Type:
application/yaml" --data-binary @modify_geotag.yaml
```

For more information, refer to Sample Geo-tagging Configuration File - site1, on page 201 and Sample Geo-tagging Configuration File - site2, on page 201.

Step 16 Add monitor database for both sites.

For example:

```
curl -i -X PATCH http://installer-site1:8458/api/system/config/application-config -H "Content-Type:
application/yaml" --data-binary @monitor_db.yaml
```

```
curl -i -X PATCH http://installer-site2:8458/api/system/config/application-config -H "Content-Type:
application/yaml" --data-binary @monitor_db2.yaml
```

For sample configuration, refer to Sample Monitor Database Configuration File - site1, on page 201 and Sample Monitor Database Configuration File - site2, on page 202.

Arbiter Installation on OpenStack

Before you begin

- Latest ISO Image
- Latest base VMDK
- Glance images
- Cinder Volumes, only for ISO (SVN and mongo are not needed) are created
- Access and Security (22 and mongo port 27717 to 27720 are opened as per deployment)

Note For more information on the above mentioned prerequisites, refer to *CPS Installation Guide for OpenStack*.

Note If you want to add the MongoDB authentication, you need to enable *dbAuthenticationEnabled* parameter. For more information refer to *Configuration Parameters - HA System* section in *CPS Installation Guide for OpenStack*. You need to mention password for all the sites separately using API and that must be same for all the sites.

Step 1 Create flavors by executing the following command:

```
nova flavor-create --ephemeral 0 arbiter auto 4096 0 2
```

Step 2 Cloud init configuration for Arbiter: When Arbiter is launched, `arbiter-cloud.cfg` file needs to be passed via user-data. In order to pass `arbiter-cloud.cfg` file, it should be placed in the directory where the user will execute `nova boot` command (likely the path will be `/root/cps-install` directory).

Create `arbiter-cloud.cfg` file with the following content:

```
#cloud-config
write_files:
 - path: /etc/sysconfig/network-scripts/ifcfg-eth0
   encoding: ascii
   content: |
     DEVICE=eth0
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.20.38.251           ---> update with your internal address
     NETMASK=255.255.255.0          ---> update with your netmask
     GATEWAY=172.20.38.1            ---> update with your gateway
     NETWORK=172.20.38.0            ---> update with your network
   owner: root:root
   permissions: '0644'
 - path: /var/lib/cloud/instance/payload/launch-params
   encoding: ascii
   owner: root:root
   permissions: '0644'
 - path: /root/.autoinstall.sh
   encoding: ascii
   content: |
     #!/bin/bash
     if [[ -d /mnt/iso ]] && [[ -f /mnt/iso/install.sh ]]; then
       /mnt/iso/install.sh << EOF
     arbiter
     y
     1
     EOF
     fi
     /root/.enable_firewall.sh
     /root/.mongo_auth.sh
     /root/.add_db_hosts.sh
     if [[ -x "/var/qps/install/current/scripts/upgrade/reinit.sh" ]]; then
         /var/qps/install/current/scripts/upgrade/reinit.sh
     fi
   permissions: '0755'
 - path: /root/.enable_firewall.sh
   encoding: ascii
   content: |
     #!/bin/bash
     mkdir -p /etc/facter/facts.d/
     cat <<EOF >/etc/facter/facts.d/qps_firewall.txt
     firewall_disabled=0            ---> change it to 1 if you do not want firewall enabled on this
 setup and remove below fields
     internal_address=172.20.38.251 ---> update with your internal address
     internal_device=0
     EOF
   permissions: '0755'
- path: /root/.mongo_auth.sh
  encoding: ascii
  content: |
    #!/bin/bash
    mkdir -p /etc/facter/facts.d/
    cat <<EOF >/etc/facter/facts.d/mongo_auth.txt
    db_authentication_enabled=FALSE     ---> if mongo-auth enable then make it to TRUE
    db_authentication_admin_passwd=     ---> provide admin user encrypted password if enable
    db_authentication_readonly_passwd=  ---> provide readonly user encrypted password if enable
    EOF
```

```
    permissions: '0755'
  - path: /root/.add_db_hosts.sh  ---> update db hosts IP as per requirement
    encoding: ascii
    content: |
      #!/bin/bash
      #Example if /etc/broadhop/mongoConfig.cfg:
      #[SESSION-SET1]
      #SETNAME=set01
      #OPLOG_SIZE=5120
      #ARBITER=arbiter-site3:27717
      #ARBITER_DATA_PATH=/var/data/sessions.1/set01
      #PRIMARY-MEMBERS
      #MEMBER1=sessionmgr01-site1:27717
      #MEMBER2=sessionmgr02-site1:27717
      #SECONDARY-MEMBERS
      #MEMBER1=sessionmgr01-site2:27717
      #MEMBER2=sessionmgr02-site2:27717
      #DATA_PATH=/var/data/sessions.1/set01
      #[SESSION-SET1-END]
      #For above mongoConfig.cfg below hosts entries are needed in /etc/hosts, edit below list as per
  your requirement
      cat <<EOF >> /etc/hosts
      192.168.1.1 arbiter-site3
      192.168.1.2 sessionmgr01-site1
      192.168.1.3 sessionmgr02-site1
      192.168.1.4 sessionmgr01-site2
      192.168.1.5 sessionmgr02-site2
      EOF
    permissions: '0755'
mounts:
  - [ /dev/vdb, /mnt/iso, iso9660, "auto,ro", 0, 0 ]
runcmd:
  - ifdown eth0
  - echo 172.20.38.251 installer arbiter >> /etc/hosts ---> update this IP
  - ifup eth0
  - /root/.autoinstall.sh
```

Note Edit IPADDR/NETMASK/NETWORK/GATEWAY and remove the hint information while using the `cloud-config` file. For example, internal network information and so on.

Step 3 Create Arbiter VM:

Note As DHCP has been disabled in the prep script, the `arbiter-cloud.cfg` file needs to be passed to the arbiter to assign IP addresses to arbiter interfaces.

Before executing `nova boot` command, confirm that the cloud configuration file (`arbiter-cloud.cfg`) exists in the right directory.

Execute the following command to create arbiter VM with two NICs:

```
source ~/keystonerc_core
nova boot --config-drive true --user-data=arbiter-cloud.cfg  --file
/root/keystonerc_user=/root/keystonerc_core
--image "base_vm" --flavor "arbiter"
--nic net-id="9c89df81-90bf-45bc-a663-e8f80a8c4543,v4-fixed-ip=172.16.2.19"
--nic net-id="dd65a7ee-24c8-47ff-8860-13e66c0c966e,v4-fixed-ip=172.18.11.101"
--block-device-mapping "/dev/vdb=eee05c17-af22-4a33-a6d9-cfa994fecbb3:::0"
--availability-zone "az-2:os24-compute-2.cisco.com" arbiter
```

For example,

```
nova boot --config-drive true --user-data=arbiter-cloud.cfg
--file /root/keystonerc_user=/root/keystonerc_core
```

```
--image "base_vm" --flavor "arbiter" --nic net-id="<Internal n/w id>,v4-fixed-ip=<Interanl n/w private
 ip>"
--nic net-id="<Management n/w id>,v4-fixed-ip=<Management n/w public ip>"
--block-device-mapping "/dev/vdb=<Volume id of iso>:::0"
--availability-zone "<availability zone:Host info>" arbiter
```

The following examples can be used to get the internal and management IP addresses and volume IDs which are used to spin arbiter VM.

```
source ~/keystonerc_core
```

neutron net-list

id	name	subnet
9c89df81-90bf-45bc-a663-e8f80a8c4543	internal	682eea79-6eb4-49db-8246-3d94087dd487 172.16.2.0/24
8d60ae2f-314a-4756-975f-93769b48b8bd	gx	9f3af6b8-4b66-41ce-9f4f-c3016154e027 192.168.2.0/24
dd65a7ee-24c8-47ff-8860-13e66c0c966e	management	a18d5329-1ee9-4a9e-85b5-c381d9c53eae 172.18.11.0/24

nova volume-list

ID	Status	Display Name	Size	Volume Type	Attached to
146ee37f-9689-4c85-85dc-c7fee85a18f4	available	mongo2	60	None	
6c4146fa-600b-41a0-b291-11180224d011	available	mongo01	60	None	
181a0ead-5700-4f8d-9393-9af8504b15d8	available	snv02	2	None	
158ec525-b48b-4528-b255-7c561c8723a9	available	snv02	2	None	
eee05c17-af22-4a33-a6d9-cfa994fecbb3	available	cps-production-7.9.9-SNAPSHOT.iso	3	None	

For the kernel upgrade, once the Arbiter deployment is complete and the output of `diagnostics.sh` command displays no errors, execute the following command from Cluster Manager to ensure that the kernel version is upgraded.

```
/var/qps/install/current/scripts/upgrade/reinit.sh
```

This command will prompt for reboot choice. Please select **Y** for the same and proceed.

Configuration Parameters - GR System

`grConfig` section under `applicationConfig` holds configuration for all GR related configurations. The following parameters can be defined in the CPS configuration file for GR system.

All parameters and values are case sensitive.

Note Before loading the configuration file to your CPS cluster, verify that the YAML file uses the proper syntax.

Various configuration files like, qns.conf, mon_db* related configuration files, gr_cluster.conf files have been modified to support GR installation using API.

- `policyServerConfig`
- `dbMonitorForQns`
- `dbMonitorForLb`
- `clusterInfo`

policyServerConfig

`policyServerConfig` holds configuration for `/etc/broahdop/qns.conf` file and supported parameters in it.

In `policyServerConfig`, a new parameter `deploymentType` has been added which is not a part of `qns.conf` file which is used for validation of `qns.conf` file parameters. It can have values for HA or GR deployments. By default, the value is set to GR. In case of GR, validation for required parameters in configuration will happen.

For the parameter descriptions, consult your Cisco Technical Representative.

Table 11: policyServerConfig Parameters

qns.conf Parameter	Corresponding Parameter in policyServerConfig
-DGeoSiteName	geoSiteName
-DSiteId	siteId
-DRemoteSiteId	remoteSiteId
-DheartBeatMonitorThreadSleepMS	heartBeatMonitorThreadSleepMS
-Dcom.mongodb.updaterConnectTimeoutMS	mongodbupdaterConnectTimeoutMS
-Dcom.mongodb.updaterSocketTimeoutMS	mongodbupdaterSocketTimeoutMS
-DdbConnectTimeout	dbConnectTimeout
-Dmongo.client.thread.maxWaitTime	threadMaxWaitTime
-DdbSocketTimeout	dbSocketTimeout
-DclusterFailureDetectionMS	clusterFailureDetectionMS
-Dremote.locking.off	remoteLockingOff
-DapirouterContextPath	apirouterContextPath

qns.conf Parameter	Corresponding Parameter in policyServerConfig
-Dua.context.path	uaContextPath
-Dcom.cisco.balance.dbs	balanceDbs
-DsprLocalGeoSiteTag	sprLocalGeoSiteTag
-DbalanceLocalGeoSiteTag	balanceLocalGeoSiteTag
-DsessionLocalGeoSiteTag	sessionLocalGeoSiteTag
-DclusterPeers	clusterPeers
-DgeoHASessionLookupType	geoHaSessionLookupType
-DisGeoHAEnabled	isGeoHaEnabled
-DmaxHash	maxHash
-DdbSocketTimeout.remoteBalance	dbSocketTimeoutRemoteBalance
-DdbConnectTimeout.remoteBalance	dbConnectTimeoutRemoteBalance
-Dmongo.connections.per.host.remoteBalance	mongoConnHostRemoteBalance
-Dmongo.threads.allowed.to.wait.for. connection.remoteBalance	waitThreadNumRemoteBalance
-Dmongo.client.thread.maxWaitTime.remoteBalance	threadWaitTimeRemoteBalance
-DdbSocketTimeout.remoteSpr	dbSocketTimeoutRemoteSpr
-DdbConnectTimeout.remoteSpr	dbConnectTimeoutRemoteSpr
-Dmongo.connections.per.host.remoteSpr	mongoConnHostRemoteSpr
-Dmongo.threads.allowed.to.wait.for.connection.remoteSp	waitThreadNumRemoteSpr
-Dmongo.client.thread.maxWaitTime.remoteSpr	threadWaitTimeRemoteSpr
-DenableReloadDictionary	enableReloadDict
-Dcom.broadhop.q.if	replicationIface
-DRemoteGeoSiteName	remoteGeoSiteName
-Dcom.broadhop.run.clusterId	clusterId

dbMonitorForQns and dbMonitorForLb

`dbMonitorForQns` holds configuration for `/etc/broadhop/mon_db_for_callmodel.conf` file and supported parameters in it.

`dbMonitorForLb` holds configuration for `/etc/broadhop/mon_db_for_lb_failover.conf` file and supported parameters in it.

```
applicationConfig:
dbMonitorForLb:
     setName:
       - "SPR-SET1"
       - "BALANCE-SET1"
       - "SESSION-SET1"
       - "ADMIN-SET1"
dbMonitorForQns:
     stopUapi: "true"
     setName:
       - "SPR-SET1"
       - "BALANCE-SET1"
       - "SESSION-SET1"
```

`monQnsLB: "true"` must be added under `config:` (`api/system/config/config`) section in YAML file to stop Policy Server (QNS) processes from lb01/lb02 when all the policy services are down (that is, qns01,02..n). Once policy server processes from lb01/lb02 go down, above configuration make sure that the traffic switchover takes place.

For `mon_db*` config, `setName` is an array of set names and corresponds to `title` in YAML for `replicaSet` configuration. The following is an example configuration:

```
---- title: "SESSION-SET1"
     setName: "set01"
     oplogSize: "1024"
     arbiter: "arbiter-site3:27717"
     arbiterDataPath: "/var/data/sessions.1"
     primaryMembers:
```

clusterInfo

`clusterInfo` section under `grConfig` holds configuration for `gr_cluster.conf` file and supported parameters in it.

YAML will have following format for clusterInfo:

- remotePcrfclient01IP: Specifies remote sites pcrfclient01 IP.
- remotePcrfclient02IP: Specifies remote sites pcrfclient02 IP.

When user specifies cluster info details, local site details are fetched from existing configuration and based on all information `gr_cluster.conf` is updated which populates admin database with cluster information.

Example Requests and Response

Retrieve Current Configuration

To retrieve (GET) the current configuration:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/config/application-config

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, refer to *HTTPS Support for Orchestration API* section in *CPS Installation Guide for OpenStack*.

- **Header:** Content-Type: application/yaml
- **Method:** GET
- **Payload:** There is no payload.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error
  - Example Response (YAML format):

```
---
policyServerConfig:
       geoSiteName: "SITE1"
       clusterId: "Cluster-SITE1"
       siteId: "SITE1"
       remoteSiteId: "SITE2"
       heartBeatMonitorThreadSleepMS: "500"
       mongodbupdaterConnectTimeoutMS: "1000"
       mongodbupdaterSocketTimeoutMS: "1000"
       dbConnectTimeout: "1200"
       threadMaxWaitTime: "1200"
       dbSocketTimeout: "600"
       remoteLockingOff: ""
       apirouterContextPath: ""
       uaContextPath: ""
       balanceDbs: ""
       clusterPeers: ""
       isGeoHaEnabled: "true"
       geoHaSessionLookupType: "realm"
       enableReloadDict: "true"
       sprLocalGeoSiteTag: "SITE1"
       balanceLocalGeoSiteTag: "SITE1"
       sessionLocalGeoSiteTag: "SITE1"
       deploymentType: "GR"
 dbMonitorForQns:
  stopUapi: "true"
  setName:
   - "SESSION-SET1"
 dbMonitorForLb:
  setName:
   - "SESSION-SET1"
```

Note In case there is an error in configuring `geoHaSessionLookupType`, CPS behaves incorrectly and drop messages. The following logs come continuously if there is an error in the configuration:

```
GeoHA is enabled, unknown lookuptype is configured:
<>. Possible values are…
```

Update Configuration

When this API call completes, the Cluster Manager configuration is updated and all new VMs are deployed asynchronously.

Note The amount of time needed to complete the process depends on the number of VMs being deployed.

Use this API to load an updated configuration on the CPS Cluster Manager: You can specify this configuration during fresh install time or also at a later stage once system is deployed using PATCH. The following information gives details about PATCH method:

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system/mongo/config/application-config

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, refer to *HTTPS Support for Orchestration API* section in *CPS Installation Guide for OpenStack*.

- **Header:** Content-Type: application/yaml
- **Method:** PATCH
- **Payload:** Include the YAML configuration file in the PATCH request. The entire contents of the configuration must be included.
- **Response Codes:** 200 OK: success; 400: The request is invalid; 500: Server Error

  Example Response (YAML format):

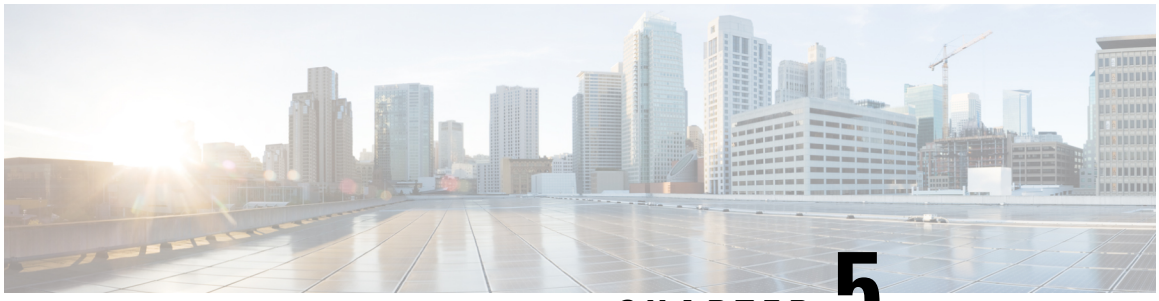
  Note After using this API to load the updated configuration, you must apply the configuration.

```
curl -i -X PATCH http://installer:8458/api/system/config/application-config -H
"Content-Type: application/yaml" --data-binary @mondblb.yaml


HTTP/1.1 200 OK
Date: Fri, 19 Aug 2016 10:31:49 GMT
Content-Length: 0
```

`cat mondblb.yaml`: The following is an example request to change mon_db* script configuration:

```
dbMonitorForLb:
      setName:
        - ADMIN-SET1
        - BALANCE-SET1
```



CHAPTER 5

Geographic Redundancy Configuration

- Database Migration Utilities, on page 65
- Recovery Procedures, on page 69
- Additional Session Replication Set on GR Active/Active Site, on page 84
- Network Latency Tuning Parameters, on page 96
- Remote SPR Lookup based on IMSI/MSISDN Prefix, on page 96
- Remote Balance Lookup based on IMSI/MSISDN Prefix, on page 98
- SPR Provisioning, on page 100
- Configurations to Handle Traffic Switchover, on page 113
- Remote Databases Tuning Parameters, on page 116
- SPR Query from Standby Restricted to Local Site only (Geo Aware Query), on page 117
- Balance Location Identification based on End Point/Listen Port, on page 120
- Balance Query Restricted to Local Site, on page 121
- Session Query Restricted to Local Site during Failover, on page 123
- Publishing Configuration Changes When Primary Site becomes Unusable, on page 126
- Graceful Cluster Shutdown, on page 128
- Active/Active Geo HA - Multi-Session Cache Port Support, on page 129
- Handling RAR Switching, on page 137
- Configure Cross-site Broadcast Messaging, on page 137
- Configure Redundant Arbiter (arbitervip) between pcrfclient01 and pcrfclient02, on page 139
- Moving Arbiter from pcrfclient01 to Redundant Arbiter (arbitervip), on page 140

Database Migration Utilities

The database migration utilities can be used to migrate a customer from Active/Standby Geographic Redundancy (GR) environment to Active/Active Geographic Redundancy environment. Currently, the migration utilities support doing remote database lookup based on NetworkId (i.e. MSISDN, IMSI, and so on). The user needs to split the SPR and balance databases from Active/Standby GR model i.e. one for each site in Active/Active GR model.

The workflow for splitting the databases is as follows:

- Dump the mongoDB data from active site of active/standby system using `mongodump` command.
- Run the Split Script, on page 66 on SPR and balance database files collected using `mongodump` command.

- Restore the mongo database for each site with `mongorestore` command using files collected from running Split Script, on page 66.

After the database splitting is done, you can audit the data by running the Audit Script, on page 68 on each set of site-specific database files separately.

The Split Script, on page 66 is a python script to split SPR and balance database into two site specific parts. The file `split.csv` is the input file which should have the Network Id regex strings for each site. The Audit Script, on page 68 is a tool to do auditing on the split database files to check for any missing/orphaned records.

To extract the database migration utility, execute the following command:

```
tar -zxvf /mnt/iso/app/install/xxx.tar.gz -C /tmp/release-train-directory
```

where, *xxx* is the release train version.

This command will extract release train into `/tmp/release-train-directory`.

Split Script

The split script first splits the SPR database into two site-specific SPR databases based on the *network_id_key* field. Then it loops through the balance database to check which site each balance record correlates to based on the *subscriberId* field and puts the balance record into one of two site-specific balance databases. If there is no match, then it is considered as `Orphaned balance record` and added to `nositebal.json`.

Here are the usage details of the split script:

Usage

```
python split.py split.csv > output.txt
```

Prerequisite

The prerequisite to run the script is `python-pymongo` module. To install python-pymongo on CPS VMs, run the command `yum install python-pymongo`.

System Requirements

- RAM: Minimum 1 GB of free memory. The script is memory-intensive and it needs at least 1 GB of RAM to work smoothly.
- vCPUs: Minimum 4 vCPUs. The script is CPU intensive.
- Persistent Storage: Free storage is required which should be at least as much as the Active/Standby database file sizes. SSD storage type is preferred (for faster runtimes) but not required.

Input Files

- The command line argument `split.csv` is a CSV file that will have network ID regex strings listed per site. The format of each line is site-name, one or more comma-separated regex strings. The regex format is python regex.

  Here is an example of a `split.csv` file where the networkId regex strings are in the MSISDN Prefix format (i.e. "Starts With" type in Policy Builder configuration).

  site1,5699[0-9],5697[0-9]*,5695[0-9]*,5693[0-9]*,5691[0-9]**

*site2,569[86420][0-9]**

Here is another example where the networkId strings are in the suffix format (i.e. "Ends With" type in Policy Builder configuration).

site1,^.[0-4]$*

site2,^.[5-9]$*

Important Since this is a CSV file, using "," in regex strings would result in unexpected behavior, so avoid using "," in regex strings.

- The script looks for the file `subscriber.bson` and one or more `account.bson` files in current directory. The `account.bson` files could be in nested folders to support a sharded balance database. The balance database could be compressed or uncompressed (the script does not look into the compressed fields).

Output Files

- `site1-balance-mgmt_account.bson`
- `site1_spr_subscriber.bson`
- `site2-balance-mgmt_account.bson`
- `site2_spr_subscriber.bson`

In addition there will be following error/debug output files:

- `errorbal.json`
- `errorspr.json`
- `nositebal.json`
- `nositespr.json`

Here is the output from a sample run of the split script.

```
$ time python split.py split.csv > output.txt
real8m44.015s
user8m0.236s
sys0m35.270s

$ more output.txt
Found the following subscriber file
./spr/spr/subscriber.bson
Found the following balance files
./balance_mgmt/balance_mgmt/account.bson
./balance_mgmt_1/balance_mgmt_1/account.bson
./balance_mgmt_2/balance_mgmt_2/account.bson
./balance_mgmt_3/balance_mgmt_3/account.bson
./balance_mgmt_4/balance_mgmt_4/account.bson
./balance_mgmt_5/balance_mgmt_5/account.bson
Site1 regex strings: 5699[0-9]*|5697[0-9]*|5695[0-9]*|5693[0-9]*|5691[0-9]*
Site2 regex strings: 569[86420][0-9]*

Started processing subscriber file
```

….

….

<snip>

Audit Script

The audit script first goes through the balance database and retrieves a list of IDs. Then it loops through each record in SPR database and tries to match the *network_id_key* or *_id* with the ID list from balance database. If there is no match, they are tagged with the counter for `Subscribers missing balance records`.

Here are the usage details for the audit script:

Usage

```
python audit.py > output.txt
```

Prerequisite

The prerequisite to run the script is `python-pymongo` module. To install python-pymongo on CPS VMs, run the command `yum install python-pymongo`.

System Requirements

- RAM: Minimum 1 GB of free memory. The script is memory-intensive and it needs at least 1 GB of RAM to work smoothly.
- vCPUs: Minimum 4 vCPUs. The script is CPU intensive.

Input Files

The script looks for the file `subscriber.bson` and one or more `account.bson` files in current directory. The `account.bson` files could be in nested folders to support a sharded balance database. The balance database could be compressed or uncompressed (the script does not look into the compressed fields).

Output Files

`sprbalmissing.bson`

Sample console output from script before splitting SPR and balance databases.

```
Total subscriber exceptions: 0
Total subscriber errors: 0
Total subscriber empty records: 1
Total subscriber records: 6743644
Total subscriber matched records: 6733102
Total subscriber missing records: 10541
```

After running the script on site-specific databases after the split, the user gets the following:

Site1:

```
Total subscriber exceptions: 0
Total subscriber errors: 0
Total subscriber empty records: 1
Total subscriber records: 4137817
```

```
Total subscriber matched records: 4131978
Total subscriber missing records: 5839
```

Site2:

```
Total subscriber exceptions: 0
Total subscriber errors: 0
Total subscriber empty records: 1
Total subscriber records: 2605826
Total subscriber matched records: 2601124
Total subscriber missing records: 4702
```

Recovery Procedures

This section covers the following recovery cases in Geographic Redundancy environment:

- Site recovery after entire site fails.
- Individual virtual machines recovery.
- Databases and replica set members recovery.

Site Recovery Procedures

Manual Recovery

When a site fails, it is assumed that other (secondary or standby) site is now operational and has become primary.

Here are the steps to recover the failed site manually:

Step 1 Confirm that the databases are in primary/secondary state on running site.

Step 2 Reset the member priorities of failed site so that when the site recovers, these members do not become primary.

a) Log on to current primary member and reset the priorities by executing the following commands:

Note To modify priorities, you must update the `members` array in the replica configuration object. The array index begins with 0. The array index value is different than the value of the replica set member's `members[n]._id` field in the array.

```
ssh <current primary replica set member>
mongo --port <port>
conf=rs.conf()
###here, note the output and note array index value of members for which we want to reset the
priorities.
#Assuming that array index value of members of failed members are 1 and 2,
conf.members[1].priority=2
conf.members[2].priority=3
conf.members[3].priority=5
conf.members[4].priority=4
rs.reconfig(conf)
#Ensure the changed priorities are reflected.
exit
```

Step 3 Re-configure gateways to make sure that no traffic is sent to failed site during recovery stage.

Step 4 Power on the VMs in the following sequence:

a) Cluster Manager
b) pcrfclients

Stop the following two scripts using monit on both pcrfclients to avoid automatic switchover of databases or automatic stopping of load balancer processes:

```
mon_db_for_call_model
```

```
mon_db_for_lb_failover
```

c) Session managers
d) Policy Server (QNS)
e) Load balancers

Step 5 Synchronize the timestamps between the sites for all VMs by executing the following command from pcrfclient01 of current secondary (recovering) site:

```
/var/qps/bin/support/sync_times.sh gr
```

Important The script should be executed only when policy director (lbs) time has been synced (NTP).

Step 6 Confirm that the databases on the failed site completely recovers and become secondary. If they do not become secondary, refer to Database Replica Members Recovery Procedures, on page 72.

Step 7 After the databases are confirmed as recovered and are secondary, reset these database's priorities using `set_priority.sh` script from Cluster Manager so that they become primary.

Step 8 If possible, run sample calls and test if recovered site is fully functional or not.

Step 9 Reconfigure the gateways to send the traffic to recovered site.

Step 10 Start `mon_db_for_call_model` and `mon_db_for_lb_failover` scripts on both pcrfclients.

Step 11 Monitor the recovered site for stability.

Automatic Recovery

CPS allows you to automatically recover a failed site.

In a scenario where a member fails to recover automatically, use the procedures described in Manual Recovery, on page 69.

For VMware

For VMware (CSV based installations), execute `automated_site_recovery.py` script on a failed site. The script recovers the failed replica members that are in RECOVERING or FATAL state. The script is located on Cluster Manager at `/var/qps/bin/support/gr_mon/automated_site_recovery.py`. The script starts the QNS processes on the Load Balancer VMs, resets the priorities of the replica set, and starts the DB monitor script. However, the script does not alter the state of the VIPs.

If you provide the replica set name as an input parameter, the script recovers the failed member of that replica set.

```
python /var/qps/bin/support/gr_mon/automated_site_recovery.py --setname <setname>
```

For example, `python /var/qps/bin/support/gr_mon/automated_site_recovery.py --setname set01`

If you do not provide any input parameter to the script, the script searches for all replica members from all sets and determines if any of the replica members are in RECOVERING or FATAL state. If yes, the script recovers the members of that replica set.

You can also execute the script with `-force`. The `-force` option recovers the replica members that are in RECOVERING, FATAL and STARTUP/STARTUP2 state as well. The script starts the QNS processes on the Load Balancer VMs in the course of recovering the DB Member. However, the script does not alter the state of the VIPs. The `-force` option must only be used when any database replica member does not come out of STARTUP/STARTUP2 state automatically.

For example: `python /var/qps/bin/support/gr_mon/automated_site_recovery.py --setname set01 --force`

During recovering of a failed site, if some replica set members do not recover, then such errors are logged in the log file located at `/var/log/broadhop/scripts/automated_site_recovery.log`.

For Open Stack

The following APIs are used to trigger a recovery script for a failed site.

The logs are located at `/var/log/orchestration-api-server.log` on the Cluster Manager VM.

/api/site/recover/start

This API is used to trigger a recovery script for a failed site. The API must only be used during a planned maintenance phase. Cluster and database processes may get reset during this process and traffic is affected. This API must only be used when the cluster is in a failed state.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/site/recover/start

  Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see chapter Installation in *CPS Installation Guide for OpenStack*.

- **Header:** Content-Type: application/json
- **Method:** POST
- **Payload:** YAML with force and setName fields

  force: true/false

  setName: All replica sets or specific replica set
- **Response:** 200 OK: success; 400 Bad Request: The input parameters are malformed or invalid.
- **Example:**

```
"force": "true "
"setName": "set01"
```

/api/system

This API is used to view the status of a recovery process.

- **Endpoint and Resource:** http://<*Cluster Manager IP*>:8458/api/system

Note If HTTPS is enabled, the Endpoint and Resource URL changes from HTTP to HTTPS. For more information, see chapter Installation in *CPS Installation Guide for OpenStack*.

- **Header:** Content-Type: application/json
- **Method:** GET
- **Payload:** No payload
- **Response:** 200 OK: success; 500: Script config not found
- **Example:**

  Recovery is currently underway

  ```
  "state":"recovering"
  ```

  or

  Problem with the recovery

  ```
  "state":"error_recovering"
  ```

Individual Virtual Machines Recovery

During recovery, the CPS VMs should come UP automatically on reboot without intervention. However, there are scenarios when the VMs will not recover, may be they are unstable or have become corrupt.

The following options exist to recover these CPS VMs:

- Reinitialize a VM — If a VM is stable but configurations are modified or have become corrupt and one is willing to restore the VM (reset all configurations, else, the configurations can be corrected manually). In that case, execute `/etc/init.d/vm-init-client` from the VM. Note that if the IP addresses are changed, this command would not recover the same.
- Redeploy a VM — If current VM is not recoverable, the operator may run the command `deploy.sh <vm-name>` from the cluster manager. This command will recreate the VM with latest saved configurations.

Database Replica Members Recovery Procedures

CPS database replica members can be recovered automatically or manually.

Automatic Recovery

A replica member holds a copy of the operations log (oplog) that is usually in synchronization with the oplog of the primary member. When the failed member recovers, it starts syncing from previous instance of oplog and recovers. If the member is session_cache whose data is on `/tmpfs` and if it is recovering from a reboot, the data and oplog has been lost. Therefore, the member resynchronizes all the data from primary's data files first, and then from the primary's oplog.

Verification: Execute `diagnostics.sh` and verify REPLICA STATE and LAST SYNC status.

- If REPLICA STATE does not come up as SECONDARY, and stuck into RECOVERING state for longer duration, then follow Manual Recovery, on page 75. (Refer Verification Step 1, on page 73)
- Also, if REPLICA STATE comes up as SECONDARY but does not catch up with PRIMARY and also you can see that replica lag is increasing (in LAST SYNC). (Refer Verification Step 2, on page 73)

Verification Step 1

Execute `diagnostics.sh` script to verify if all the members are healthy.

```
diagnostics.sh --get_replica_status <sitename>

CPS Diagnostics GR Multi-Node Environment
--------------------------
Checking replica sets...
|-------------------------------------------------------------------------------------------------------------|
| Mongo:3.2.10                          MONGODB REPLICA-SETS STATUS INFORMATION OF site1
           Date : 2017-02-20 16:35:30 |
|-------------------------------------------------------------------------------------------------------------|
|  SET NAME  -  PORT  : IP ADDRESS       - REPLICA STATE -           HOST NAME
 -  HEALTH  -   LAST SYNC   - PRIORITY |
|-------------------------------------------------------------------------------------------------------------|
| BALANCE:set10
                                        |
|  Member-1  -  27718 : 172.20.18.54     -  ARBITER      -  arbiter-site3
 - ON-LINE  -   --------    -    0     |
|  Member-2  -  27718 : 172.20.17.40     -  RECOVERING   -  sessionmgr01-site1
 - ON-LINE  -   10 sec      -    2     |
|  Member-3  -  27718 : 172.20.17.38     -  RECOVERING   -  sessionmgr02-site1
 - ON-LINE  -   10 sec      -    3     |
|  Member-4  -  27718 : 172.20.19.29     -  PRIMARY      -  sessionmgr01-site2
 - ON-LINE  -    --------   -    5     |
|  Member-5  -  27718 : 172.20.19.27     -  SECONDARY    -  sessionmgr02-site2
 - ON-LINE  -    0 sec      -    4     |
|-------------------------------------------------------------------------------------------------------------|
```

Verification Step 2

1. Execute `diagnostics.sh` script to verify if all the members are healthy.

```
diagnostics.sh --get_replica_status <sitename>

CPS Diagnostics GR Multi-Node Environment
--------------------------
Checking replica sets...
|-------------------------------------------------------------------------------------------------------------|
| Mongo:3.2.10                          MONGODB REPLICA-SETS STATUS INFORMATION OF site1
              Date : 2017-02-20 16:35:30 |
|-------------------------------------------------------------------------------------------------------------|
|  SET NAME  -  PORT  : IP ADDRESS       - REPLICA STATE -           HOST NAME
    -  HEALTH  -   LAST SYNC   - PRIORITY |
|-------------------------------------------------------------------------------------------------------------|
| BALANCE:set10
                                         |
|  Member-1  -  27718 : 172.20.18.54     -  ARBITER      -  arbiter-site3
    - ON-LINE  -   --------    -    0     |
|  Member-2  -  27718 : 172.20.17.40     -  SECONDARY    -  sessionmgr01-site1
    - ON-LINE  -    10 sec     -    2     |
|  Member-3  -  27718 : 172.20.17.38     -  SECONDARY    -  sessionmgr02-site1
    - ON-LINE  -    10 sec     -    3     |
|  Member-4  -  27718 : 172.20.19.29     -  PRIMARY      -  sessionmgr01-site2
```

```
    - ON-LINE  -    --------   -    5     |
|  Member-5  -  27718 : 172.20.19.27     -  SECONDARY    -  sessionmgr02-site2
    - ON-LINE  -    0 sec      -    4     |
|-------------------------------------------------------------------------------|
```

2. Execute the following command from mongo CLI (PRIMARY member) to verify if replica lag is increasing:

```
mongo sessionmgr01-site2:27718

set10:PRIMARY> rs.printSlaveReplicationInfo()
```

If it is observed that the lag is increasing, then run `rs.status` to check for any exception. Refer to Manual Recovery, on page 75 to fix this issue.

```
mongo —-host sessionmgr01-site2 port 27718

set10:PRIMARY> rs.status()
{
"set" : "set10",
"date" : ISODate("2017-02-15T07:21:35.367Z"),
"myState" : 2,
"term" : NumberLong(-1),
"heartbeatIntervalMillis" : NumberLong(2000),
"members" : [
{
"_id" : 0,
"name" : "arbiter-site3:27718",
"health" : 1,
"state" : 7,
"stateStr" : "ARBITER",
"uptime" : 28,
"lastHeartbeat" : ISODate("2017-02-15T07:21:34.612Z"),
"lastHeartbeatRecv" : ISODate("2017-02-15T07:21:34.615Z"),
"pingMs" : NumberLong(150),
"configVersion" : 2566261
},
{
"_id" : 1,
"name" : "sessionmgr01-site1:27718",
"health" : 1,
"state" : 2,
"stateStr" : "SECONDARY",
"uptime" : 28,
"optime" : Timestamp(1487066061, 491),
"optimeDate" : ISODate("2017-02-14T09:54:21Z"),
"lastHeartbeat" : ISODate("2017-02-15T07:21:34.813Z"),
"lastHeartbeatRecv" : ISODate("2017-02-15T07:21:34.164Z"),
"pingMs" : NumberLong(0),
"lastHeartbeatMessage" : "could not find member to sync from",
"configVersion" : 2566261
},
{
"_id" : 2,
"name" : "sessionmgr02-site1:27718",
"health" : 1,
"state" : 2,
"stateStr" : "SECONDARY",
"uptime" : 30,
"optime" : Timestamp(1487066061, 491),
"optimeDate" : ISODate("2017-02-14T09:54:21Z"),
"infoMessage" : "could not find member to sync from",
"configVersion" : 2566261,
```

```
"self" : true
},
{
"_id" : 3,
"name" : "sessionmgr01-site2:27718",
"health" : 1,
"state" : 1,
"stateStr" : "PRIMARY",
"uptime" : 28,
"optime" : Timestamp(1487145333, 99),
"optimeDate" : ISODate("2017-02-15T07:55:33Z"),
"lastHeartbeat" : ISODate("2017-02-15T07:21:34.612Z"),
"lastHeartbeatRecv" : ISODate("2017-02-15T07:21:34.603Z"),
"pingMs" : NumberLong(150),
"electionTime" : Timestamp(1487066067, 1),
"electionDate" : ISODate("2017-02-14T09:54:27Z"),
"configVersion" : 2566261
},
{
"_id" : 4,
"name" : "sessionmgr02-site2:27718",
"health" : 1,
"state" : 2,
"stateStr" : "SECONDARY",
"uptime" : 28,
"optime" : Timestamp(1487145333, 95),
"optimeDate" : ISODate("2017-02-15T07:55:33Z"),
"lastHeartbeat" : ISODate("2017-02-15T07:21:34.613Z"),
"lastHeartbeatRecv" : ISODate("2017-02-15T07:21:34.599Z"),
"pingMs" : NumberLong(150),
"syncingTo" : "sessionmgr01-site2:27718",
"configVersion" : 2566261
}
],
"ok" : 1
}
```

In a scenario where a member fails to recover automatically, the operator should use procedures described in Manual Recovery, on page 75.

Manual Recovery

Before performing recovery steps, refer to the following guidelines:

- Perform the recovery steps in a maintenance window on any production system. These recovery steps require restarts of the application.
- If a failure impacts the system for a long period (for example, data center, power or hardware failure) the database instance must be resynchronized manually as the oplog will have rolled over. Full resynchronizations of the database are considered events that operation teams should execute during maintenance windows with personnel monitoring the status of the platform.
- In Geo Redundancy, replica sets are used for different databases. The replication happens based on oplog. The oplog is a data structure that mongo maintains internally at Primary where the data operations logs are maintained. The secondaries fetch from the oplog entries from the primary asynchronously and apply those operations on themselves to achieve synchronization. If a secondary goes down for a long time, due to the limited size of oplog, there is a chance that some of the logs in oplog will be overwritten by new entries. In that case, when secondary comes up, it is unable to synchronize with the primary as it does not see a timestamp from where it had gone down.

Therefore, manual resynchronization is required which is termed as initial-sync in MongoDB. In this scenario, mongod process is stopped on concerned secondary, all the data in data directory is deleted and mongod process is started. The secondary Session Manager first copies all the data from primary and then copies the oplog.

Note These procedures are only for manually recovering the databases. Based on the system status (all VMs down, traffic on other site, LBs down or up, all session cache down or only one down etc.), execution of some pre- and post- tests may be required. For example, if only one session manager is to be recovered, and we have primary database and traffic on current site, we must not reset the database priorities.

Similarly, if all of the CPS databases and load balancers are down, monit processes corresponding to `mon_db_for_lb_failover` and `mon_db_for_call_model` scripts should be stopped. These scripts monitor load balancers and if LB processes or LB itself are down, they make local instances of databases secondary. Also, if local databases are secondary, these scripts shut down load balancer process. All these scripts refer to corresponding configurations in `/etc/broadhop`. Also, post recovery, user can run some sample calls on recovered site to make sure that system is stable, and then finally migrate traffic back to original Primary.

This following sections provide the detailed steps to recover a MongoDB when the database replica set member does not recover by itself:

Recovery Using Repair Option

The repair option can be used when few members have not recovered due to VM reboot or abrupt VM shutdown or some other problem.

Step 1 Execute the diagnostics script (on pcrfclient01/02) to know which replica set or respective member has failed.

For Site1:

```
#diagnostics.sh --get_replica_status site1
```

For Site2:

```
#diagnostics.sh --get_replica_status site2
```

Step 2 Log onto session manager VM and check if mongod process is running or not.

```
#ps -ef | grep 27720
```

Note Port number can be different.

Step 3 If process is running, then shut down the process and try to repair the database.

a) To stop the process.

```
/etc/init.d/sessionmgr-<port#> stop
```

b) To repair database.

```
/etc/init.d/sessionmgr-<port#> repair
```

Sometimes the repair process takes time to recover. Check the mongo log to find the status:

```
#tailf /var/log/mongodb-<port#>.log
```

c) If the repair process is completed successfully, then start mongo process.

```
/etc/init.d/sessionmgr-<port#> start
```

Step 4 Execute the diagnostics script again (on pcrfclient01/02) to know if replica set member has recovered.

For Site1:

```
#diagnostics.sh --get_replica_status site1
```

For Site2:

```
#diagnostics.sh --get_replica_status site2
```

Step 5 To recover other failed members, follow the recovery steps from Step 1, on page 76 to Step 4, on page 77.

If the secondary member is still in RECOVERING state, refer Recovery Using Remove/Add Members Option, on page 77.

Recovery Using Remove/Add Members Option

Remove Specific Members

Caution Before removing the particular member from the replica set, make sure that you have identified correct member.

Sometimes a member lags behind significantly due to failure or network issues, and is unable to resync. In such cases, remove that member from replica set and add it again so that it can resync from start and come up.

Step 1 Log in to Cluster Manager.

Step 2 Execute the diagnostic script to know which replica set or respective member needs to be removed.

For Site1:

```
#diagnostics.sh --get_replica_status site1
```

For Site2:

```
#diagnostics.sh --get_replica_status site2
```

Step 3 Execute `build_set.sh` with the port option to remove particular member from replica set. Script prompts to enter *<VM>:<port>* where member resides.

```
cd /var/qps/bin/support/mongo/
```

For session database:

```
#./build_set.sh --session --remove-members
```

For SPR database:

```
#./build_set.sh --spr --remove-members
```

Step 4 Execute the diagnostic script again to verify if that particular member is removed.

For Site1:

```
#diagnostics.sh --get_replica_status site1
```

For Site2:

```
#diagnostics.sh --get_replica_status site2
```

Add Members

To add the earlier removed members to replica set, perform the following steps:

Step 1 Log in to Cluster Manager.

Step 2 Execute the diagnostic script to know which replica set member is not in configuration or failed member.

For Site1:

```
#diagnostics.sh --get_replica_status site1
```

For Site2:

```
#diagnostics.sh --get_replica_status site2
```

Step 3 Execute `build_set.sh` with the following option to add member(s) into replica set. This operation adds the members across the site, which were removed earlier.

```
#cd /var/qps/bin/support/mongo/
```

For session database:

```
#./build_set.sh --session --add-members

Replica-set Configuration
--------------------------------------------------------------------------------
  Please select your choice for replica sets
             sharded (1) or non-sharded (2): 2
```

For SPR database:

```
#./build_set.sh --spr --add-members

Replica-set Configuration
--------------------------------------------------------------------------------
  Please select your choice for replica sets
             sharded (1) or non-sharded (2): 2
```

Step 4 Set priority using `set_priority.sh` command. The following are example commands:

```
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db session
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db spr
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db admin
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db balance
```

Step 5 Execute the diagnostic script from Cluster Manager to know if member(s) are added successfully into the replica set.

For Site1:

```
#diagnostics.sh --get_replica_status site1
```

For Site2:

```
#diagnostics.sh --get_replica_status site2
```

Recovery for High TPS

When the HA/GR setup is running with high TPS and if replica members are having high latency between them then some replica members can go to RECOVERING state and will not recover from that state unless some commands are executed to recover those replica members. We can use the manual/automated recovery procedure to recover the replica members which are in RECOVERING state.

Automated Recovery

There can be three different scenarios of setup possible for recovery of replica members:

1. Case 1: Two members of replica set are in RECOVERING state
2. Case 2: With all replica members except primary are in RECOVERING state
3. Case 3: Some replica members are in RECOVERING state

Note Automation script recovers only those replica members which are in RECOVERING state.

Step 1 Before executing automated recovery script (`high_tps_db_recovery.sh` <*replica_setname*>), go to current primary member (site-1) and reset the priorities by executing the following commands:

Note To modify priorities, you must update the `members` array in the replica configuration object. The array index begins with 0. The array index value is different than the value of the replica set member's `members[n]._id` field in the array.

```
ssh <primary member>
mongo --port <port>
conf=rs.conf()
conf.members[1].priority=2
conf.members[2].priority=3
conf.members[3].priority=5
conf.members[4].priority=4
rs.reconfig(conf)
exit
```

Step 2 Execute the following command script to recover the member:

`high_tps_db_recovery.sh` <*replica_setname*>

For Example:

```
high_tps_db_recovery.sh SPR-SET1
```

Step 3 Execute `diagnostics.sh` command to check whether the RECOVERING member has recovered.

```
diagnostics.sh --get_replica_status
```

After the replica set member is recovered, the state will change to SECONDARY and all the process logs are stored in a log file.

Note If you are unable to recover the replica set member from RECOVERING state using automated recovery script, refer to Manual Recovery, on page 80.

Manual Recovery

Step 1 Before recovery, on all concerned replica-set VMs, perform the following steps:

a) Edit `sshd_config` file.

```
vi /etc/ssh/sshd_config
```

b) Add the following entry at the end of `sshd_config` file. The below value (130) should be based on number of files that we have under secondary's data directory. It should be close to the number of files there.

```
MaxStartups 130
```

c) Restart sshd service by executing the following command:

```
service sshd restart
```

d) Execute `diagnostics.sh --get_replica_status` command to know which members are down.

Based on the status of system and subject to above note, check if you need to reset member priorities. For example, if site-1 is Primary and site-2 is Secondary and if site-1 has gone down, we need to login to new Primary and reset the replica members priorities in such a way that when site-1 comes UP again, it would not become Primary automatically.

For this purpose, perform the following steps:

1. Go to current primary member (site-1) and reset the priorities by executing the following commands:

   Note To modify priorities, you must update the `members` array in the replica configuration object. The array index begins with 0. The array index value is different than the value of the replica set member's `members[n]._id` field in the array.

```
ssh <primary member>
mongo --port <port>
conf=rs.conf()
conf.members[1].priority=2
conf.members[2].priority=3
conf.members[3].priority=5
conf.members[4].priority=4
rs.reconfig(conf)
exit
```

2. Also on the failed replica set site there are chances that monitoring scripts would have stopped the load balancers, and shifted all the databases primaries to other site. Stop the monitoring on the failed site pcrfclient01 and pcrfclient02 both by executing the following commands:

```
monit stop mon_db_for_lb_failover
monit stop mon_db_for_callmodel
```

   At this point the operator should maintain two consoles: one for executing commands to recover the secondary member and another to manage the source secondary. The source Secondary is the Secondary that is nearest in terms of latency from the recovering Secondary.

Also, note down the port and data directory for these members. Typically these are the same, and only the host name will be different.

Step 2 Recover the member:

a) Go to recovering Secondary and execute the following commands:

```
ssh <recovering Secondary>
ps -eaf | grep mongo
/etc/init.d/sessionmgr-<port> stop
cd <member data directory>
\rm -rf *
cp /var/qps/bin/support/gr_mon/fastcopy.sh
```

b) Go to nearest working available secondary and execute the following commands:

```
ssh <source Secondary> mongo --port <mongo port>
#lock this secondary from writes
db.fsyncLock()
exit
ps -eaf | grep mongo
cd <data directory>
tar -cvf _tmp.tar _tmp
```

Any errors can be ignored.

```
tar -cvf rollback.tar rollback
```

c) Go to recovering Secondary and execute the following commands:

```
cd <data directory>
./fastcopy.sh <nearest working secondary>
<secondary data directory>
ps -eaf | grep scp | wc -l
```

d) After the count is one, start secondary by executing the following commands:

```
tar -xvf _tmp.tar
tar -xvf rollback.tar
/etc/init.d/sessionmgr-<port> start
```

e) Go to nearest secondary from where you are recovering and execute the following commands:

```
mongo --port <port>
db.fsyncUnlock()
db.printSlaveReplicationInfo()
```

Exit the database by executing `exit` command.

Monitor the lag for some time (close to 3 to 4 minutes). Initially the lag will be small, later it will increase and then decrease. This is because secondary is catching up with primary oplog and also calculating the lag. As the secondary has just restarted, it takes sometime to calculate real lag, but MongoDB shows intermittent values, hence, we see the lag initially increasing. On the other hand, the secondary is also catching up on synchronization and eventually the lag would reduce to one second or less. The member should become secondary soon.

On similar lines, recover another secondary, preferably from the just recovered one if it is closest in terms of ping connectivity.

Once all the secondaries are recovered, we need to reset the priorities and then restart the stopped load balancers.

f) Connect to primary of concerned replica set:

```
ssh <primary member>
mongo --port <port>
conf=rs.conf()
```

Based on the output, carefully identify the correct members and their ids:

Note To modify priorities, you must update the `members` array in the replica configuration object. The array index begins with 0. The array index value is different than the value of the replica set member's `members[n]._id` field in the array.

```
conf.members[1].priority=5
conf.members[2].priority=4
conf.members[3].priority=3
conf.members[3].priority=2
rs.reconfig(conf)
exit
```

Step 3 Log in to lb01 and lb02 and start monit for all qns processes.

Step 4 Check the status of qns processes on each of load balancers VMs by executing the following command:

```
monit status qnsXX
```

Step 5 Now login to pcrfclient01 and 02 and start the monit scripts that we had stopped earlier.

```
monit start mon_db_for_lb_failover
monit start mon_db_for_callmodel
```

Now reset the sshd connection on all virtual machines. Comment out the **MaxStartup** line in `/etc/ssh/sshd_conf` file and restart sshd using `service sshd restart` command.

Rebuild Replica Set

There could be a situation when all replica set members go into recovery mode and none of members are either in primary or secondary state.

Step 1 Stop CPS processes.

Step 2 Log in to Cluster Manager.

Step 3 Execute the diagnostic script to know which replica set (all members) have failed.

For Site1:

```
#diagnostics.sh --get_replica_status site1
```

For Site2:

```
#diagnostics.sh --get_replica_status site2
```

The output displays which replica set members of replica set set01 for session data are in bad shape.

Step 4 Build session replica sets. Select 2 for session non-sharded sets from Cluster Manager.

```
#cd /opt/broadhop/installer/support/mongo/
# ./build_set.sh --session --create --setname set01
Starting Replica-Set Creation
Please select your choice: replica sets sharded (1) or non-sharded (2):

2
```

Step 5 Set priority using `set_priority.sh` command. The following are example commands:

```
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db session
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db spr
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db admin
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db balance
```

Step 6 To recover other failed set, follow the recovery steps from Step 1, on page 82 to Step 4, on page 82 above.

Step 7 Restart CPS.

```
restartall.sh
```

Add New Members to the Replica Set

Step 1 Log in to Cluster Manager.

Step 2 Execute the diagnostic script to know which replica-set member is not in configuration or failed member.

For Site1:

```
#diagnostics.sh --get_replica_status site1
```

For Site2:

```
#diagnostics.sh --get_replica_status site2
```

Step 3 Update the member information in the `mongoConfig.cfg` file.

Example: The following is the example for adding two members in balance replica-set.

```
cd /etc/broadhop
vi mongoConfig.cfg
```

Before Update	**After Update**
[BALANCE-SET1] SETNAME=set03	[BALANCE-SET1] SETNAME=set03
ARBITER=pcrfclient01-prim-site-1:37718	ARBITER=pcrfclient01-prim-site-1:37718
ARBITER_DATA_PATH=/data/sessions.3	ARBITER_DATA_PATH=/data/sessions.3
PRIMARY-MEMBERS	PRIMARY-MEMBERS
MEMBER1=sessionmgr01-site-1:27718	MEMBER1=sessionmgr01-site-1:27718
SECONDARY-MEMBERS	MEMBER2=sessionmgr02-site-1:27718
MEMBER1=sessionmgr01-site-2:27718	SECONDARY-MEMBERS
DATA_PATH=/data/sessions.3	MEMBER1=sessionmgr01-site-2:27718
[BALANCE-SET1-END]	MEMBER2=sessionmgr02-site-2:27718
	DATA_PATH=/data/sessions.3

	[BALANCE-SET1-END]

Example to execute `build_set.sh` with below option to add member/s into replica-set.

```
cd /var/qps/bin/support/mongo/
```

Example: For balance database

```
./build_set.sh --balance --add-new-members Starting Replica-Set Creation
Please select your choice: replica sets sharded (1) or non-sharded (2):

2
```

Step 4 Execute the diagnostic script to know if member/s are added successfully into the replica-set.

For Site1:

```
#diagnostics.sh --get_replica_status site1
```

For Site2:

```
#diagnostics.sh --get_replica_status site2
```

Additional Session Replication Set on GR Active/Active Site

The objective of this section is to add one additional session replication set in GR Active/Active site.

The steps mentioned in this section needs to be executed from primary site Cluster Manager.

Note The steps in this section assume that the engineer performing the steps has SSH and VMware vCenter access to the production PCRF System. No impact to traffic is foreseen during the implementation of the steps although there might be a slight impact on response time during rebalance CLI.

Before you begin

You should run all the sanity checks prior and after executing the steps in this section.

Step 1 Run `diagnostics.sh` to verify that the system is in healthy state.

Step 2 Log in to primary Cluster Manager using SSH.

Step 3 Take the backup of `/etc/broadhop/mongoConfig.cfg` file.

```
cp /etc/broadhop/mongoConfig.cfg /etc/broadhop/mongoConfig.cfg.date.BACKUP
```

Step 4 Take the backup of admin database from Cluster Manager.

```
[root@cm-a ~]# mkdir admin
[root@cm-a ~]# cd admin
[root@cm-a admin]# mongodump -h sessionmgr01 --port 27721
connected to: sessionmgr01:27721
2016-09-23T16:31:13.962-0300 all dbs
** Truncated output **
```

Step 5 Edit `/etc/broadhop/mongoConfig.cfg` file using `vi` editor. Find the section for session replication set. Add the new session replication set members.

Note Server name and ports are specific to each customer deployment. Make sure that new session replication set has unique values.

Session set number must be incremented. Make sure to provide a new data path otherwise `build_set.sh` deletes the existing data and that causes an outage.

Make sure the port used in MEMBER1, MEMBER2, MEMBER3, MEMBER4, and so on are same.

```
#SITE1_START
[SESSION-SET2]
SETNAME=set10
OPLOG_SIZE=1024
ARBITER=pcrfclient01a:27727
ARBITER_DATA_PATH=/var/data/sessions.1/set10
MEMBER1=sessionmgr01a:27727
MEMBER2=sessionmgr02a:27727
MEMBER3=sessionmgr01b:27727
MEMBER4=sessionmgr02b:27727
DATA_PATH=/var/data/sessions.1/set10
[SESSION-SET2-END]

#SITE2_START
[SESSION-SET5]
SETNAME=set11
OPLOG_SIZE=1024
ARBITER=pcrfclient01b:47727
ARBITER_DATA_PATH=/var/data/sessions.1/set11
MEMBER1=sessionmgr01b:37727
MEMBER2=sessionmgr02b:37727
MEMBER3=sessionmgr01a:37727
MEMBER4=sessionmgr02a:37727
DATA_PATH=/var/data/sessions.1/set11
[SESSION-SET5-END]
```

Note Verify that the `/etc/hosts` file on the both sites is correctly configured with alias.

Site1 `/etc/hosts` file should have the following content:

```
x.x.x.a sessionmgr01 sessionmgr01a
x.x.x.b sessionmgr02 sessionmgr02a
y.y.y.a psessionmgr01 sessionmgr01b
y.y.y.b psessionmgr02 sessionmgr02b
```

Site2 `/etc/hosts` file should have the following content:

```
y.y.y.a sessionmgr01 sessionmgr01b
y.y.y.b sessionmgr02 sessionmgr02b
x.x.x.a psessionmgr01 sessionmgr01a
x.x.x.b psessionmgr02 sessionmgr02a
```

Step 6 SSH to Cluster-A/Site1 Cluster Manager. Run `build_set` to create new session replication set from Cluster Manager.

```
build_set.sh --session --create --setname set10
```

```
Replica-set Configuration
--------------------------------------------------------------------------------
REPLICA_SET_NAME: set10
  Please select your choice for replica sets
```

```
             sharded (1) or non-sharded (2): 2

WARNING: Continuing will drop mongo database and delete everything under
  /var/data/sessions.1/set10 on all Hosts
CAUTION: This result into loss of data

Are you sure you want to continue (y/yes or n/no)? : yes

  Replica-set creation                      [  Done  ]

The progress of this script can be monitored in the following log:
/var/log/broadhop/scripts/build_set_02062016_220240.log
--------------------------------------------------------------------------------
```

Step 7 Set priority using `set_priority.sh` command. The following are example commands:

```
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db session
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db spr
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db admin
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db balance
```

Step 8 Add shard to Cluster-B/Site2. This can be run from Cluster-A (Cluster Manager).

```
build_set.sh --session --create --setname set11
```

```
Replica-set Configuration
--------------------------------------------------------------------------------
REPLICA_SET_NAME: set11
  Please select your choice for replica sets
             sharded (1) or non-sharded (2): 2

WARNING: Continuing will drop mongo database and delete everything under
  /var/data/sessions.1/set11 on all Hosts
CAUTION: This result into loss of data

Are you sure you want to continue (y/yes or n/no)? : yes

  Replica-set creation                      [  Done  ]

 The progress of this script can be monitored in the following log:
/var/log/broadhop/scripts/build_set_02062016_230127.log
--------------------------------------------------------------------------------
```

Step 9 Set priority using `set_priority.sh` command. The following are example commands:

```
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db session
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db spr
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db admin
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db balance
```

Step 10 Run `build_etc.sh` to make sure modified `mongoConfig.cfg` file is restored after the reboot.

```
/var/qps/install/current/scripts/build/build_etc.sh
```

```
Building /etc/broadhop...
Copying to /var/qps/images/etc.tar.gz...
Creating MD5 Checksum...
```

Step 11 Copy `mongoConfig.cfg` file to all the nodes using `copytoall.sh` from Cluster Manager.

```
copytoall.sh /etc/broadhop/mongoConfig.cfg /etc/broadhop/mongoConfig.cfg
```

```
Copying '/var/qps/config/mobile/etc/broadhop/mongoConfig.cfg'
to '/etc/broadhop/mongoConfig.cfg' on all VMs
lb01
mongoConfig.cfg

      100% 4659     4.6KB/s   00:00
lb02
mongoConfig.cfg

      100% 4659     4.6KB/s   00:00
sessionmgr01
** Truncated output **
```

Step 12 Transfer the modified `mongoConfig.cfg` file to Site2 (Cluster-B).

```
scp /etc/broadhop/mongoConfig.cfg cm-b:/etc/broadhop/mongoConfig.cfg


root@cm-b's password:
mongoConfig.cfg

      100% 4659     100% 4659     4.6KB/s   00:00
```

Step 13 SSH Cluster-B (Cluster Manager). Run `build_etc.sh` to make sure modified `mongoConfig.cfg` file is restored after the reboot.

```
/var/qps/install/current/scripts/build/build_etc.sh


Building /etc/broadhop...
Copying to /var/qps/images/etc.tar.gz...
Creating MD5 Checksum...
```

Step 14 Copy `mongoConfig.cfg` file from Cluster-B (Cluster Manager) to all the nodes using `copytoall.sh` from Cluster Manager.

```
copytoall.sh /etc/broadhop/mongoConfig.cfg /etc/broadhop/mongoConfig.cfg


Copying '/var/qps/config/mobile/etc/broadhop/mongoConfig.cfg'
to '/etc/broadhop/mongoConfig.cfg' on all VMs
lb01
mongoConfig.cfg

      100% 4659     100% 4659     4.6KB/s   00:00
lb02
mongoConfig.cfg

      100% 4659     100% 4659     4.6KB/s   00:00
** Truncated output **
```

Step 15 (Applicable for HA and Active/Standby GR only) Adding shards default option. Login to OSGi mode and add the shards as follows:

```
telnet qns01 9091
```

```
Trying XXX.XXX.XXX.XXX...
Connected to qns01.
Escape character is '^]'.
```

```
addshard seed1[,seed2] port db-index siteid [backup]

osgi> addshard sessionmgr01,sessionmgr02 27727 1 Site1

osgi> addshard sessionmgr01,sessionmgr02 27727 2 Site1

osgi> addshard sessionmgr01,sessionmgr02 27727 3 Site1

osgi> addshard sessionmgr01,sessionmgr02 27727 4 Site1

osgi> addshard sessionmgr01,sessionmgr02 37727 1 Site1

osgi> addshard sessionmgr01,sessionmgr02 37727 2 Site1

osgi> addshard sessionmgr01,sessionmgr02 37727 3 Site1

osgi> addshard sessionmgr01,sessionmgr02 37727 4 Site1

osgi> rebalance

osgi> migrate

Migrate ...
All versions up to date - migration starting
```

Step 16 Verify that the sessions have been created in the newly created replication set and are balanced.

```
session_cache_ops.sh --count site2

session_cache_ops.sh --count site1
```

Sample output:

```
Session cache operation script
Thu Jul 28 16:55:21 EDT 2016
-----------------------------------------------------
Session Replica-set SESSION-SET4
-----------------------------------------------------
Session Database          : Session Count
-----------------------------------------------------
 session_cache            : 1765
 session_cache_2          : 1777
 session_cache_3          : 1755
 session_cache_4          : 1750
-----------------------------------------------------
 No of Sessions in SET4   : 7047
-----------------------------------------------------

-----------------------------------------------------
Session Replica-set SESSION-SET5
-----------------------------------------------------
Session Database          : Session Count
-----------------------------------------------------
 session_cache            : 1772
 session_cache_2          : 1811
 session_cache_3          : 1738
 session_cache_4          : 1714
-----------------------------------------------------
 No of Sessions in SET5   : 7035
-----------------------------------------------------
```

Step 17 (Applicable for Active/Active GR only) Add shards with Site option. Login to OSGi mode and add the shards as follows:

Note This process is adding shards for Active/Active GR setup which has Site options enabled. To enable Geo-HA feature, refer to Active/Active Geo HA - Multi-Session Cache Port Support, on page 129.

```
telnet qns01 9091
Trying XXX.XXX.XXX.XXX...
Connected to qns01.
Escape character is '^]'.
```

Run `listsitelookup` if you are unsure about the site names. Similar information can be obtained from `/etc/broadhop/qns.conf` file (`-DGeoSiteName=Site1`).

```
osgi> listsitelookup
                  Id        PrimarySiteId       SecondarySiteId    LookupValues
                   1            Site1               Site2       pcef-gx-1.cisco.com
                   1            Site1               Site2       pcef-gy-1.cisco.com
                   2            Site2               Site1       pcef2-gx-1.cisco.com
                   2            Site2               Site1       pcef2-gy-1.cisco.com
```

Note Do not run `addshard` command on multiple sites in parallel. Wait for the command to finish on one site and then proceed to second site.

Adding shard to Site1. Run the following command from the qns of Site1:

```
osgi> addshard sessionmgr01,sessionmgr02 27727 1 Site1

osgi> addshard sessionmgr01,sessionmgr02 27727 2 Site1

osgi> addshard sessionmgr01,sessionmgr02 27727 3 Site1

osgi> addshard sessionmgr01,sessionmgr02 27727 4 Site1
```

Adding shards to Site2. Run the following command from the qns of Site2:

```
osgi> addshard sessionmgr01,sessionmgr02 37727 1 Site2

osgi> addshard sessionmgr01,sessionmgr02 37727 2 Site2

osgi> addshard sessionmgr01,sessionmgr02 37727 3 Site2

osgi> addshard sessionmgr01,sessionmgr02 37727 4 Site2
```

Run `osgi> rebalance Site1` command from Site1 qns.

Run `osgi> rebalance Site2` command from Site2 qns.

Run the following command from the Site1 qns:

```
osgi> migrate Site1
Migrate ...
All versions up to date - migration starting
```

Run the following command from the Site2 qns:

```
osgi> migrate Site2
Migrate ...
All versions up to date - migration starting
```

Step 18 Verify that the sessions have been created in the newly created replication set and are balanced.

```
session_cache_ops.sh --count site2

session_cache_ops.sh --count site1
```

Sample output:

```
Session cache operation script
Thu Jul 28 16:55:21 EDT 2016
-----------------------------------------------------
Session Replica-set SESSION-SET4
-----------------------------------------------------
Session Database          : Session Count
-----------------------------------------------------
 session_cache            : 1765
 session_cache_2          : 1777
 session_cache_3          : 1755
 session_cache_4          : 1750
-----------------------------------------------------
 No of Sessions in SET4   : 7047
-----------------------------------------------------

-----------------------------------------------------
Session Replica-set SESSION-SET5
-----------------------------------------------------
Session Database          : Session Count
-----------------------------------------------------
 session_cache            : 1772
 session_cache_2          : 1811
 session_cache_3          : 1738
 session_cache_4          : 1714
-----------------------------------------------------
 No of Sessions in SET5   : 7035
-----------------------------------------------------
```

Step 19 Secondary Key Ring Configuration: This step only applies If you are adding additional session replication set to a new session manager server. Assuming that existing setup has the secondary key rings configured for existing session Replication servers.

Refer to the section *Secondary Key Ring Configuration* in *CPS Installation Guide for VMware*.

Step 20 Configure session replication set priority from Cluster Manager.

```
cd /var/qps/bin/support/mongo/; ./set_priority.sh --db session
```

Step 21 Verify whether the replica set status and priority is set correctly by running the following command from Cluster Manager:

```
diagnostics.sh --get_replica_status

|-----------------------------------------------------------------------------------------------------------------|
| SESSION:set10
                |
|  Member-1  -  27727 : 192.168.116.33   -  ARBITER      -  pcrfclient01a    - ON-LINE  -   --------
    -    0     |
|  Member-2  -  27727 : 192.168.116.71   -  PRIMARY      -  sessionmgr01a    - ON-LINE  -   --------
    -    5     |
|  Member-3  -  27727 : 192.168.116.24   -  SECONDARY    -  sessionmgr02a    - ON-LINE  -    0 sec
      -    4     |
|  Member-4  -  27727 : 192.168.116.70   -  SECONDARY    -  sessionmgr01b    - ON-LINE  -    0 sec
      -    3     |
|  Member-5  -  27727 : 192.168.116.39   -  SECONDARY    -  sessionmgr02b    - ON-LINE  -    0 sec
      -    2     |
|-----------------------------------------------------------------------------------------------------------------|
```

Step 22 Run `diagnostics.sh` to verify whether the priority for new replication set has been configured or not.

Step 23 Add session geo tag in MongoDBs. Repeat these steps for both session replication sets.

For more information, refer to Session Query Restricted to Local Site during Failover, on page 123 for more details.

Site1 running log: This procedure only applies if customer have local site tagging enabled.

Note To modify priorities, you must update the `members` array in the replica configuration object. The array index begins with 0. The array index value is different than the value of the replica set member's `members[n]._id` field in the array.

```
mongo sessionmgr01:27727
MongoDB shell version: 2.6.3
connecting to: sessionmgr01:27727/test

set10:PRIMARY> conf = rs.conf();
{
 "_id" : "set10",
 "version" : 2,
 "members" : [
  {
   "_id" : 0,
   "host" : "pcrfclient01a:27727",
   "arbiterOnly" : true
  },
  {
   "_id" : 1,
   "host" : "sessionmgr01a:27727",
   "priority" : 5
  },
  {
   "_id" : 2,
   "host" : "sessionmgr02a:27727",
   "priority" : 4
  },
  {
   "_id" : 3,
   "host" : "sessionmgr01b:27727",
   "priority" : 3
  },
  {
   "_id" : 4,
   "host" : "sessionmgr02b:27727",
   "priority" : 2
  }
 ],
 "settings" : {
  "heartbeatTimeoutSecs" : 1
 }
}
set10:PRIMARY> conf.members[1].tags = { "sessionLocalGeoSiteTag": "Site1" }
{ "sessionLocalGeoSiteTag" : "Site1" }
set10:PRIMARY> conf.members[2].tags = { "sessionLocalGeoSiteTag": "Site1"}
{ "sessionLocalGeoSiteTag" : "Site1" }
set10:PRIMARY> conf.members[3].tags = { "sessionLocalGeoSiteTag": "Site2"}
{ "sessionLocalGeoSiteTag" : "Site2" }
set10:PRIMARY> conf.members[4].tags = { "sessionLocalGeoSiteTag": "Site2"}
{ "sessionLocalGeoSiteTag" : "Site2" }
set10:PRIMARY> rs.reconfig(conf);
{ "ok" : 1 }
```

```
set10:PRIMARY> rs.conf();
{
 "_id" : "set10",
 "version" : 3,
 "members" : [
  {
   "_id" : 0,
   "host" : "pcrfclient01a:27727",
   "arbiterOnly" : true
  },
  {
   "_id" : 1,
   "host" : "sessionmgr01a:27727",
   "priority" : 5,
   "tags" : {
    "sessionLocalGeoSiteTag" : "Site1"
   }
  },
  {
   "_id" : 2,
   "host" : "sessionmgr02a:27727",
   "priority" : 4,
   "tags" : {
    "sessionLocalGeoSiteTag" : "Site1"
   }
  },
  {
   "_id" : 3,
   "host" : "sessionmgr01b:27727",
   "priority" : 3,
   "tags" : {
    "sessionLocalGeoSiteTag" : "Site2"
   }
  },
  {
   "_id" : 4,
   "host" : "sessionmgr02b:27727",
   "priority" : 2,
   "tags" : {
    "sessionLocalGeoSiteTag" : "Site2"
   }
  }
 ],
 "settings" : {
  "heartbeatTimeoutSecs" : 1
 }
}
set10:PRIMARY>
```

Site2 TAG configuration:

Note To modify priorities, you must update the `members` array in the replica configuration object. The array index begins with 0. The array index value is different than the value of the replica set member's `members[n]._id` field in the array.

```
mongo sessionmgr01b:37727
MongoDB shell version: 2.6.3
connecting to: sessionmgr01b:37727/test
set11:PRIMARY> conf = rs.conf();
{
 "_id" : "set11",
 "version" : 2,
```

```
 "members" : [
  {
   "_id" : 0,
   "host" : "pcrfclient01b:47727",
   "arbiterOnly" : true
  },
  {
   "_id" : 1,
   "host" : "sessionmgr01b:37727",
   "priority" : 5
  },
  {
   "_id" : 2,
   "host" : "sessionmgr02b:37727",
   "priority" : 4
  },
  {
   "_id" : 3,
   "host" : "sessionmgr01a:37727",
   "priority" : 3
  },
  {
   "_id" : 4,
   "host" : "sessionmgr02a:37727",
   "priority" : 2
  }
 ],
 "settings" : {
  "heartbeatTimeoutSecs" : 1
 }
}
set11:PRIMARY> conf.members[1].tags = { "sessionLocalGeoSiteTag": "Site2"}
{ "sessionLocalGeoSiteTag" : "Site2" }
set11:PRIMARY> conf.members[2].tags = { "sessionLocalGeoSiteTag": "Site2"}
{ "sessionLocalGeoSiteTag" : "Site2" }
set11:PRIMARY> conf.members[3].tags = { "sessionLocalGeoSiteTag": "Site1"}
{ "sessionLocalGeoSiteTag" : "Site1" }
set11:PRIMARY> conf.members[4].tags = { "sessionLocalGeoSiteTag": "Site1"}
{ "sessionLocalGeoSiteTag" : "Site1" }
set11:PRIMARY> rs.reconfig(conf);
{ "ok" : 1 }
set11:PRIMARY> rs.conf();
{
 "_id" : "set11",
 "version" : 3,
 "members" : [
  {
   "_id" : 0,
   "host" : "pcrfclient01b:47727",
   "arbiterOnly" : true
  },
  {
   "_id" : 1,
   "host" : "sessionmgr01b:37727",
   "priority" : 5,
   "tags" : {
    "sessionLocalGeoSiteTag" : "Site2"
   }
  },
  {
   "_id" : 2,
   "host" : "sessionmgr02b:37727",
   "priority" : 4,
   "tags" : {
```

```
    "sessionLocalGeoSiteTag" : "Site2"
   }
  },
  {
   "_id" : 3,
   "host" : "sessionmgr01a:37727",
   "priority" : 3,
   "tags" : {
    "sessionLocalGeoSiteTag" : "Site1"
   }
  },
  {
   "_id" : 4,
   "host" : "sessionmgr02a:37727",
   "priority" : 2,
   "tags" : {
    "sessionLocalGeoSiteTag" : "Site1"
   }
  }
 ],
 "settings" : {
  "heartbeatTimeoutSecs" : 1
 }
}
set11:PRIMARY>
```

Step 24 Run `diagnostics.sh` to verify that the system is in healthy state.

Rollback Additional Session Replication Set

Caution Removing session replication set from running Production system can impact in session loss hence it is not recommended. But if there are no other options due to any circumstances, follow these instructions.

Step 1 Run `diagnostics.sh` from OAM (pcrfclient) or Cluster Manager to verify the system is in healthy state.

Step 2 Restore ADMIN database from the backup taken in Step Step 4, on page 84. Restore sharding database only.

```
mongorestore --drop --objcheck --host sessionmgr01 --port 27721 --db sharding sharding
```

Step 3 Run `restartall.sh` to restart the system.

Note If it is a GR site, run `restartall.sh` on both sites before proceeding to the next step.

Step 4 Drop the newly created session replication set. In this example, remove set15.

```
build_set.sh --session --remove-replica-set --setname set15
```

Step 5 Verify sharding errors are not reported by qns nodes. Login to pcrfclient01 of Site-1 and Site-2.

```
tailf /var/log/broadhop/consolidated-qns.log
```

Ignore the following errors:

```
2016-10-05 11:45:01,446 [pool-3-thread-1] WARN  c.b.c.m.dao.impl.ShardInterface.? - Unexpected error
java.lang.NullPointerException: null
```

```
at
com.broadhop.cache.mongodb.dao.impl.ShardInterface$MonitorShards.monitorSessionTypeStatisticsCounter(ShardInterface.java:496)
 ~[com.broadhop.policy.geoha.cache_8.1.1.r090988.jar:na]
at com.broadhop.cache.mongodb.dao.impl.ShardInterface$MonitorShards.run(ShardInterface.java:407)
~[com.broadhop.policy.geoha.cache_8.1.1.r090988.jar:na]
at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511) [na:1.8.0_45]
at java.util.concurrent.FutureTask.runAndReset(FutureTask.java:308) [na:1.8.0_45]
at
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$301(ScheduledThreadPoolExecutor.java:180)
 [na:1.8.0_45]
at
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:294)
 [na:1.8.0_45]
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142) [na:1.8.0_45]
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617) [na:1.8.0_45]
at java.lang.Thread.run(Thread.java:745) [na:1.8.0_45]
```

Step 6 Run `diagnostics.sh` from OAM (pcrfclient) or Cluster Manager to verify the system is in healthy state.

Step 7 Edit `mongoConfig.cfg` file and remove the entries related to set15 from Site-1 (Cluster-A).

Step 8 Copy `mongoConfig.cfg` file to all the nodes using `copytoall.sh` script from Cluster Manager.

```
copytoall.sh /etc/broadhop/mongoConfig.cfg /etc/broadhop/mongoConfig.cfg
```

```
Copying '/var/qps/config/mobile/etc/broadhop/mongoConfig.cfg' to '/etc/broadhop/mongoConfig.cfg'
on all VMs
lb01
mongoConfig.cfg
100%  4659  4.6KB/s  00:00
lb02
mongoConfig.cfg
100%  4659  4.6KB/s  00:00
sessionmgr01
<output truncated>
```

Step 9 Transfer the modified `mongoConfig.cfg` file to Site2 (Cluster-B).

```
scp /etc/broadhop/mongoConfig.cfg cm-b:/etc/broadhop/mongoConfig.cfg
root@cm-b's password:
mongoConfig.cfg
100%  4659  4.6KB/s  00:00
[root@cm-a ~]#
```

Step 10 SSH to Cluster-B Cluster Manager. Run `build_etc.sh` to make sure modified `mongoConfig.cfg` file is restored after reboot.

```
/var/qps/install/current/scripts/build/build_etc.sh
Building /etc/broadhop...
Copying to /var/qps/images/etc.tar.gz...
Creating MD5 Checksum...
```

Step 11 Copy `mongoConfig.cfg` file from Cluster-B Cluster Manager to all the nodes using `copytoall.sh` from Cluster Manager.

```
copytoall.sh /etc/broadhop/mongoConfig.cfg /etc/broadhop/mongoConfig.cfg
```

```
Copying '/var/qps/config/mobile/etc/broadhop/mongoConfig.cfg' to '/etc/broadhop/mongoConfig.cfg'
on all VMs
lb01
mongoConfig.cfg
100%  4659  4.6KB/s  00:00
lb02
mongoConfig.cfg
```

```
100%  4659  4.6KB/s  00:00
<output truncated>
```

Network Latency Tuning Parameters

In GR, if the network latency between two sites is more than the threshold value, `-DringSocketTimeOut`, `-DshardPingerTimeoutMs` and `-balancePingerTimeoutMs` parameters need to be configured with the appropriate values.

To get the values that must be configured in `-DringSocketTimeOut`, `-DshardPingerTimeoutMs` and `-balancePingerTimeoutMs`, check the latency using ping command for the sessionmgr which hosts the shard.

Example:

If the network latency between two sites is 150 ms, the value must be configured as 50 + 150 (network latency in ms) = 200 ms.

The parameters need to be added in `/etc/broadhop/qns.conf` on both sites:

```
-DringSocketTimeOut=200
-DshardPingerTimeoutMs=200
-DbalancePingerTimeoutMs=200
```

If the parameters are not configured, default values will be considered:

```
-DringSocketTimeOut=50
-DshardPingerTimeoutMs=75
-DbalancePingerTimeoutMs=75
```

In addition to the above parameters, the following parameters need to updated if network latency is found to be 150 ms. These values need to be increased so that QNS processes can come up and get connected to Mongo Database on Session Managers without having timeouts.

```
-Dmongo.client.thread.maxWaitTime=1700
 -Dmongo.client.thread.maxWaitTime.balance=1700
 -Dmongo.client.thread.maxWaitTime.remoteBalance=1700
 -Dmongo.client.thread.maxWaitTime.remoteSpr=1700
 -Dmongo.client.thread.maxWaitTime.cdrrep=1700
 -Dmongo.client.thread.maxWaitTime.cdr=1700
```

Remote SPR Lookup based on IMSI/MSISDN Prefix

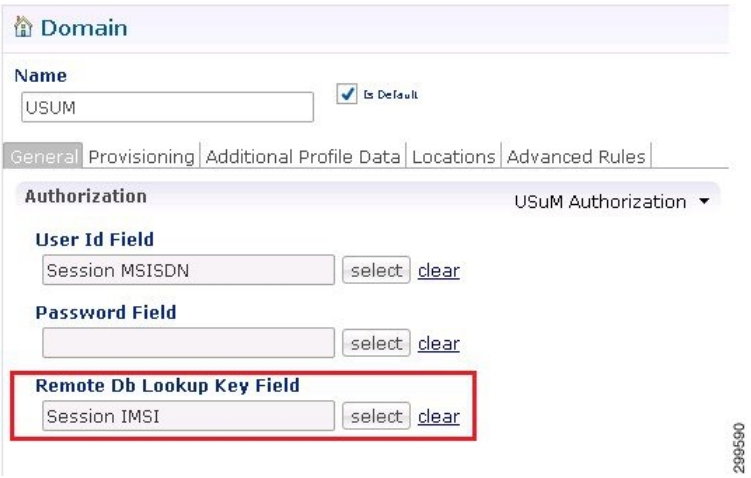
Prerequisites

Policy Builder configuration on both the sites should be the same.

Configuration

Step 1 Configure the Lookup key field in Policy Builder under 'Domain'. It can be IMSI, MSISDN, Session User Name, and so on. An example configuration is given below:

Figure 21: Remote Db Lookup Key



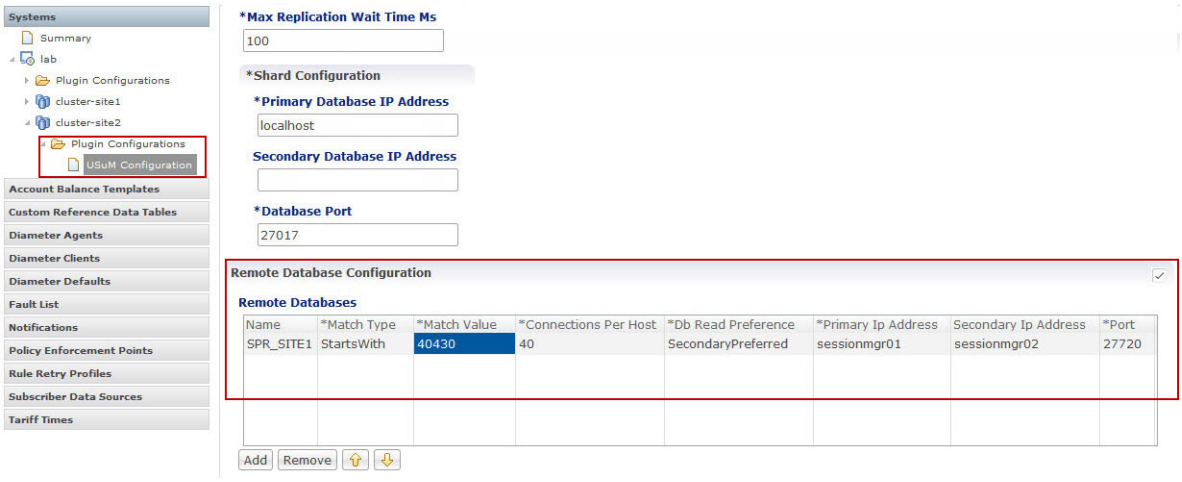
Step 2 Configure remote databases in Policy Builder under USuM Configuration.

Consider there are two sites: Site1 (Primary) and Site2 (Secondary). In Policy Builder there will be two clusters for Site1 and Site2 in case of Active/Active model.

Under 'Cluster-Site2', create USuM Configuration and add remote databases to be accessed when Site1 is not available.

An example configuration is shown below:

Figure 22: Remote Database Configuration



Parameter	Description
Name	Unique name to identify the remote database. **Note** Remote database name should be same as site name configured in `-DGeoSiteName` in `/etc/broadhop/qns.conf` file. This is needed to see the correct sites's subscriber in Control Center, when multiple SPR is configured.
Match Type	Pattern match type. It can be Starts With, Ends With, Equals, Regex match type.
Match Value	Key/regex to be used in pattern matching.
Connection per host	Number of connections that can be created per host.
Db Read Preference	Database read preferences.
Primary Ip Address, Secondary Ip Address, Port	Connection parameter to access database. This should be accessible from Site2 irrespective of Site1 is UP or DOWN.

For more information on Remote Database Configuration parameters, refer to the *CPS Mobile Configuration Guide* for this release.

Remote Balance Lookup based on IMSI/MSISDN Prefix

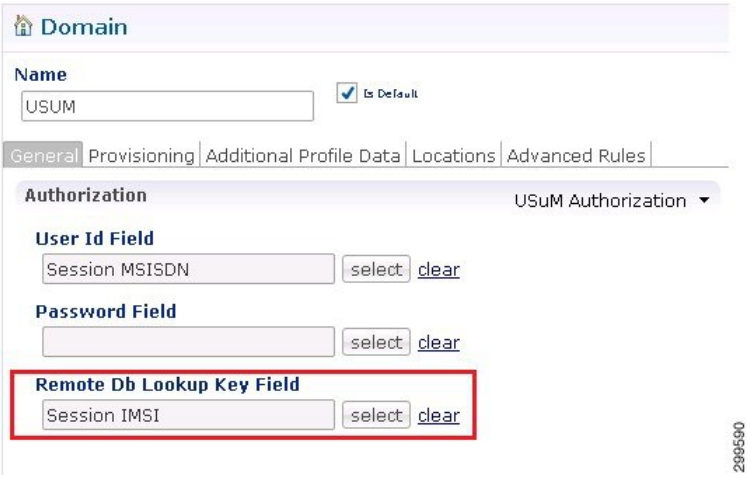
Prerequisites

Policy Builder configuration on both the sites should be the same.

Configuration

Step 1 Configure the Lookup key field in policy builder under **Domain**. It can be IMSI, MSISDN, Session User name and so on. An example configuration is given:

Figure 23: Lookup Key



Step 2 Configure remote databases in policy builder under **Balance Configuration**.

Consider there are two sites: Site1 (Primary) and Site2 (Secondary). So in Policy Builder there will be two clusters for Site1 and Site2.

Under 'Cluster-Site2', create **Balance Configuration** and add remote databases to be accessed when Site1 is not available.

An example configuration is given:

Figure 24: Example Configuration

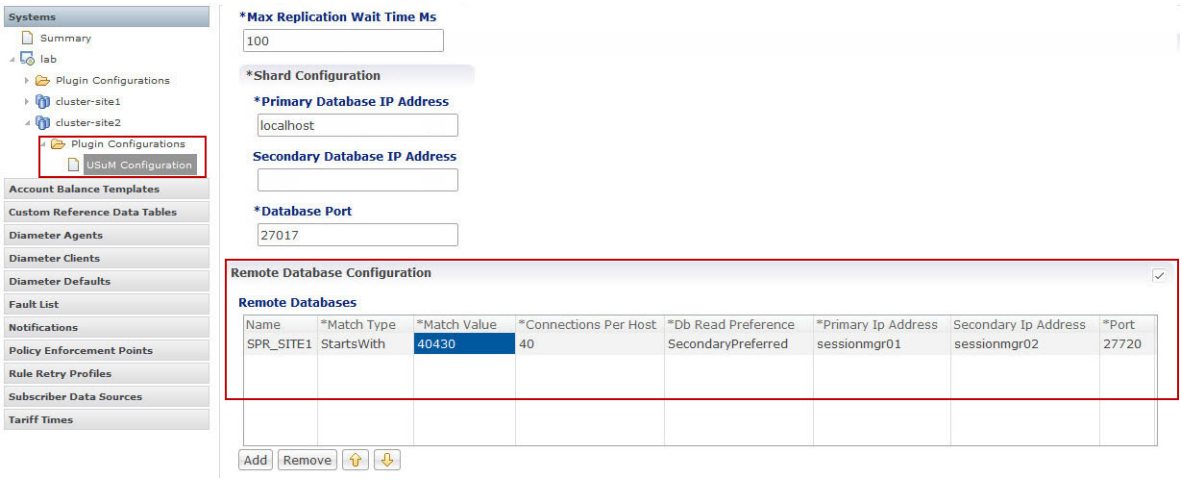


Table 12: Remote Database Configuration Parameters

Parameter	Description
Name	Unique name to identify the remote database.
Match Type	Match type to be matched for the remote database to be selected for lookup.
Connection per host	Number of connections that can be created per host.

Parameter	Description
Db Read Preference	Database read preferences.
Primary Ip Address, Secondary Ip Address, Port	Connection parameter to access database. This should be accessible from Site2 irrespective of Site1 is UP or DOWN.

For more information on **Balance Configuration** parameters, refer to the *CPS Mobile Configuration Guide* for this release.

SPR Provisioning

CPS supports multiple SPR and multiple balance databases in different ways based on deployments. SPR provisioning can be done either based on end point/listen port or using API router configuration.

SPR Location Identification based on End Point/Listen Port

Prerequisites

Policy Builder configuration on both the sites should be the same.

Configuration

Consider there are two sites: Site1 (Primary) and Site2 (Secondary).

Add new entry on Site2 in haproxy.cfg (`/etc/haproxy/haproxy.cfg`) file listening on port 8081 (or any other free port can be used) with custom header “RemoteSprDbName”. Same configuration to be done on both load balancers.

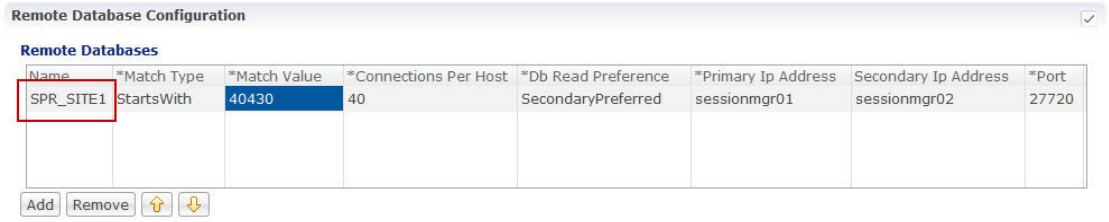
```
listen pcrf_a_proxy lbvip01:8081
        mode http
        reqadd RemoteSprDbName:\ SPR_SITE1
        balance roundrobin
        option httpclose
        option abortonclose
        option httpchk GET /ua/soap/KeepAlive
        server qns01_A qns01:8080 check inter 30s
        server qns02_A qns02:8080 check inter 30s
        server qns03_A qns03:8080 check inter 30s
        server qns04_A qns04:8080 check inter 30s
# If there are more qns add all entries here
```

Where,

RemoteSprDbName is the custom header.

SPR_SITE1 is the remote database name configured in Step 2 of Configuration, on page 97. Refer to screen below:

Figure 25: Remote Database



API Router Configuration

The following are the three use cases where a user must configure API router:

- Multiple SPR/Multiple Balance
- Common SPR/Multiple Balance (Hash Based)
- Common SPR Database and Multiple Balance Database based on SPR AVP

Use Cases

Multiple SPR/Multiple Balance

Use Case

This will be useful when there are multiple active sites and each has their own separate SPR and balance database.

Logic

- When API router receives the request it will extract the Network Id (MSISDN) from the request.
- It will iterate through the API router criteria table configured in **Api Router Configuration** in Policy Builder and find SPR, Balance database name by network Id.
- API router will make unified API call adding SPR, balance database name in http header.
- SPR and balance module will use this SPR, balance database name and make queries to appropriate databases.

Common SPR/Multiple Balance (Hash Based)

Use Case

This will be useful when there is common SPR across multiple sites, and latency between site is less. Also this will evenly distribute the data across all balance databases.

Logic

- When API router receives the create subscriber request:

- It first generates hash value using network Id (in range 0 to n-1, where n is from qns.conf parameter -DmaxHash=2)
- It will add generated hash value in SPR AVP (with code _balanceKeyHash).
- For example, if maxHash is 2 and generated hash value is 1, then SPR AVP will be _balanceKeyHash=1

- When API router receives the request other than create subscriber.
  - It will query subscriber and get hash value from subscriber AVP (_balanceKeyHash).
- Once hash value is available, it will iterate through the API router criteria table configured in **Api Router Configuration** in Policy Builder and find balance database name by hash.
- API router will make unified API call adding balance database name in http header.
- Balance module will use this balance database name and make query to appropriate balance database.

Common SPR Database and Multiple Balance Database based on SPR AVP

Use Case

This will be useful when there is common SPR across multiple sites, but each has separate balance in each site. We can add region AVP in each subscriber to read from local database.

Logic

- When API router receives the request:
  - It will query subscriber and get subscriber AVP from subscriber. AVP name is configurable.
  - Once AVP value is available, it will iterate through the API router criteria table configured in **Api Router Configuration** in Policy Builder and find balance database name by AVP.
  - API router will make unified API call adding balance database name in http header.
- Balance module will use this balance database name and make query to appropriate balance database.

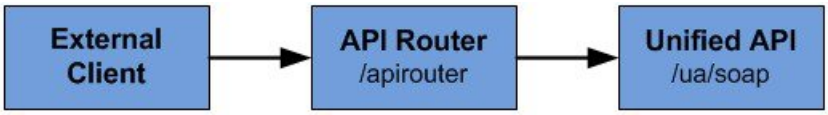
HTTP Endpoint

By default, API router is exposed on `/apirouter` and the Unified API endpoint is exposed on `/ua/soap`. The default URLs are as follows:

Table 13: Default URLs

Setup	Unified API	API Router
AIO	http://lbvip01:8080/ua/soap	http://lbvip01:8080/apirouter
HA	https://lbvip01:8443/ua/soap	https://lbvip01:8443/apirouter

Figure 26: API Router Configuration



Some customer may have configured the URL at multiple places and do not want to change URL. To change the endpoint so that the API router uses `/ua/soap`, add the following parameters in `/etc/broadhop/qns.conf`. This makes API router act as unified API.

```
-DapirouterContextPath=/ua/soap
-Dapi.ua.context.path=/ua/backend
-Dua.context.path=/ua/backend
```

Note The `api.ua.context.path` and `ua.context.path` parameters must be the same.

Figure 27: Unified API Router Configuration

New URLs are as follows:

Table 14: New URLs

Setup	Unified API	API Router
AIO	http://lbvip01:8080/ua/backend	http://lbvip01:8080/ua/soap
HA	https://lbvip01:8443/ua/backend	https://lbvip01:8443/ua/soap

Configuration

By default, API router configuration feature is not installed. To install this feature, put the following entries in feature files.

Table 15: Feature File Changes

Feature File	Feature Entry
`/etc/broadhop/pcrf/feature`	com.broadhop.apirouter.service.feature
`/etc/broadhop/pb/feature`	com.broadhop.client.feature.apirouter

Note Change in feature file requires to run `buildall.sh`, `reinit.sh` from Cluster Manager for the features to get installed.

Policy Builder Configuration

Domain

1. Remote Db lookup key field:

   This retriever fetches the value that can be used in patten match to get remote SPR, balance database name.

   This field retriever can be:

   - MSISDN retriever, IMSI retriever or whatever is networkid in subscriber to support multiple SPR, multiple balance
   - NetworkIdHash retriever to support common SPR, multiple balance based on hashing
   - Subscriber retriever AVP to support common SPR, multiple balance based on subscriberAVP

API Router Configuration

1. Enable Multi-Credential Management: This check box is used to add the support for the multi-credential management for the API router to handle the following Unified API requests: CreateSubscriber, DeleteSubscriber, GetSubscriber, CreateBalance, DeleteBalance, ChangeCredentialUsername.

Note AddCredential, AddCredentials, DeleteCredential, and DeleteCredentials API requests will be added in the future to complete the multi-credential handling.

2. Enable Backup Cache Lookup For Primary Key: This check box is used to turn on/off the step to iterate over the know databases to lookup the subscriber primary key if the secondary key cache does not find the record.

Note A CreateSubscriberRequest does not look into the cache or iterate over the databases because it inserts a new record in the database and must build the cache entries from the request.

3. Filter Type: Type of filter to be used.
   - NetworkId — To configure multiple SPR, multiple balance.
   - NetworkIdHash — To configure common SPR, multiple balance based on hashing.
   - SubscriberAVP — To configure common SPR, multiple balance based on susbcriberAVP.

     AVP Name can be changed by adding flag -DbalanceKeyAvpName=avpName. Refer to Configurable Flags, on page 106.

4. Router Criteria: The following is the list of criteria to consider for pattern matching.

Table 16: Router Criteria

Criteria	Description
Match Type	Pattern match type. It can be Starts With, Ends With, Equals, Regex match type.
Match Value	Key/regex to be used in pattern matching.
Remote SPR DB name	If criteria match, use this database name as SPR database name. This database name should match with the remote database name configured in **USuMConfiguration > Remote databases > Name**.
Remote Balance DB name	If criteria match, use this database name as Balance database name. This database name should match with the remote database name configured in **BalanceConfiguration > Remote databases > Name**.

Balance Configuration (remote databases)

The following parameters can be configured under Remote Database for Balance Configuration.

Table 17: Remote Database Parameters

Parameter	Description
Name	Balance database name
Match Type	Pattern match type It can be Starts With, Ends With, Equals, Regex match type
Match Value	Key/regex to be used in pattern matching
Connection per host	Balance database mongo connection per host
Db Read preference	Balance database read preference
Primary Ip Address	Balance database primary member IP address
Secondary Ip Address	Balance database secondary member IP address
Port	Balance database primary member IP address
Following fields needs to be configured if hot standby for balance database is needed	
Backup DB Host	Backup balance database primary member IP address
Backup DB Secondary Host	Backup balance database secondary member IP address
Backup DB Port	Backup balance database primary member IP address

USuM Configuration (remote databases)

The following parameters can be configured under Remote Database for USuM Configuration.

Table 18: Remote Database Parameters

Parameter	Description
Name	SPR database name
Match Type	Pattern match type It can be Starts With, Ends With, Equals, Regex match type
Match Value	Key/regex to be used in pattern matching
Connection per host	SPR database mongo connection per host
Db Read preference	SPR database read preference
Primary Ip Address	SPR database primary member IP address
Secondary Ip Address	SPR database secondary member IP address
Port	SPR database primary member IP address

Configurable Flags

The following flags are configurable in `/etc/broadhop/qns.conf` file:

Table 19: Configurable Flags

Flag	Description	Default Value
balanceKeyAvpName	Subscriber AVP name to be used for subscriber based multiple balance databases	_balanceKey
balanceKeyHashAvpName	Internal AVP name being used for hash based multiple balance databases	_balanceKeyHash
maxHash	Maximum value of hash to generate	No default value
ua.context.path	Unified API context path	/ua/soap
apirouterContextPath	API router context path	/apirouter

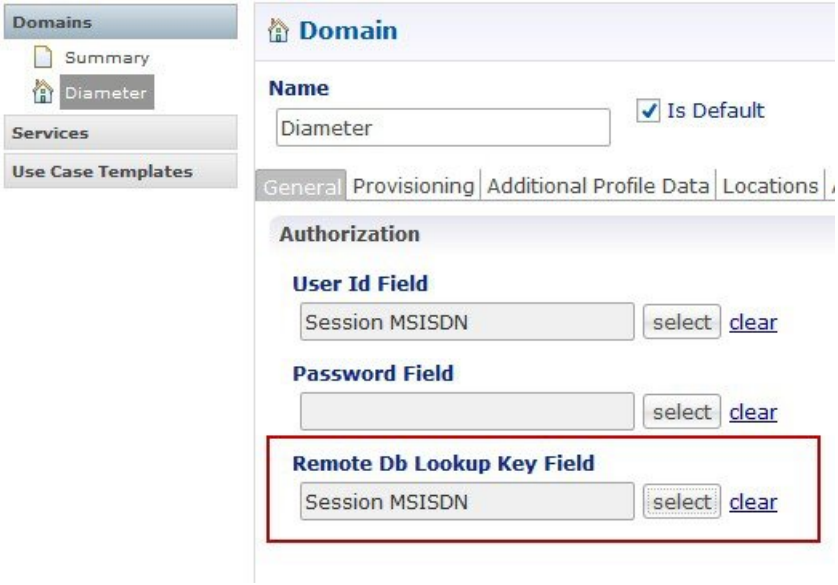
Configuration Examples

Multiple SPR/Multiple Balance

Domain Configuration

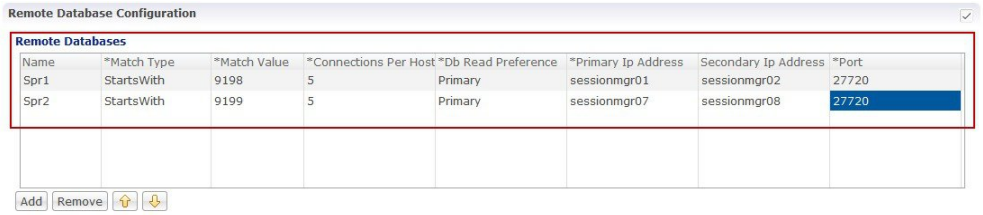
The Remote Db Lookup Key Field can be MSISDN, IMSI, and so on which is used as network ID in SPR.

Figure 28: Domain Configuration



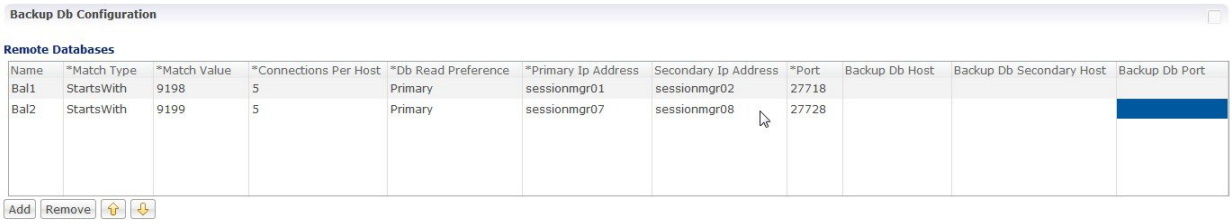
USuM Configuration

Figure 29: USuM Configuration



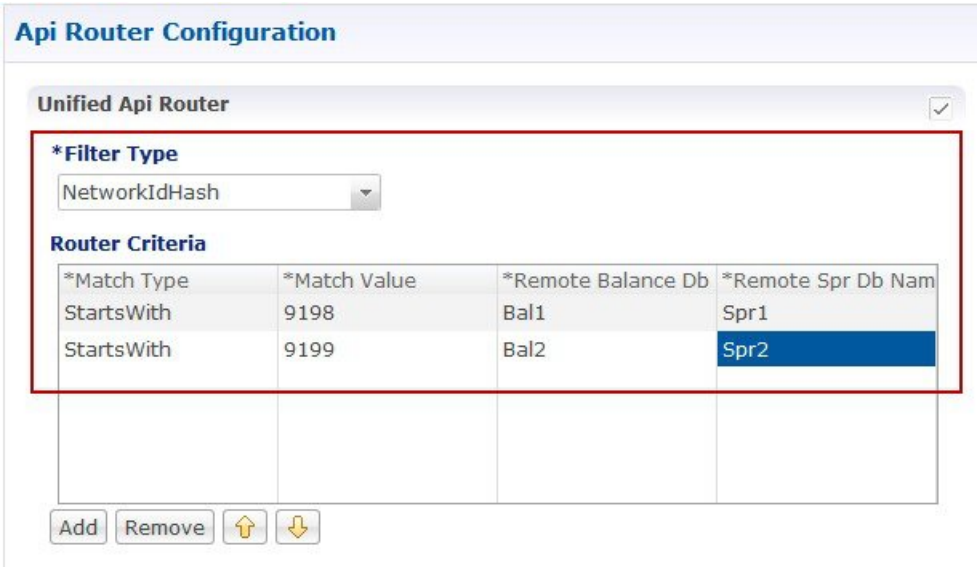
Balance Configuration

Figure 30: Balance Configuration



API Router Configuration

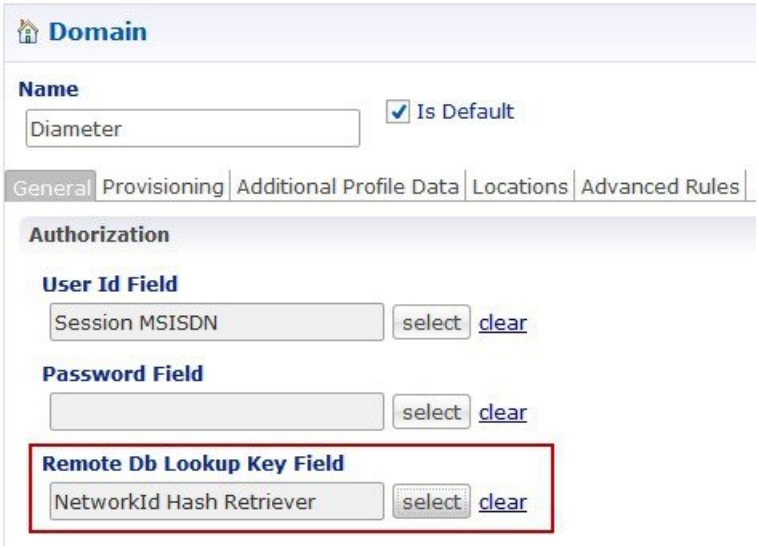
Figure 31: API Router Configuration



Common SPR/Multiple Balance (Hash Based)

Domain Configuration

Figure 32: Domain Configuration

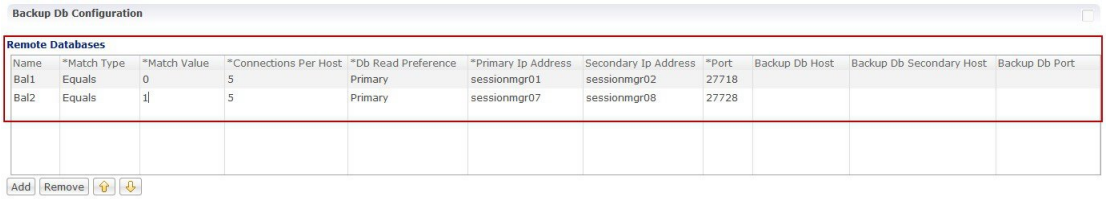


USuM Configuration

Figure 33: USuM Configuration

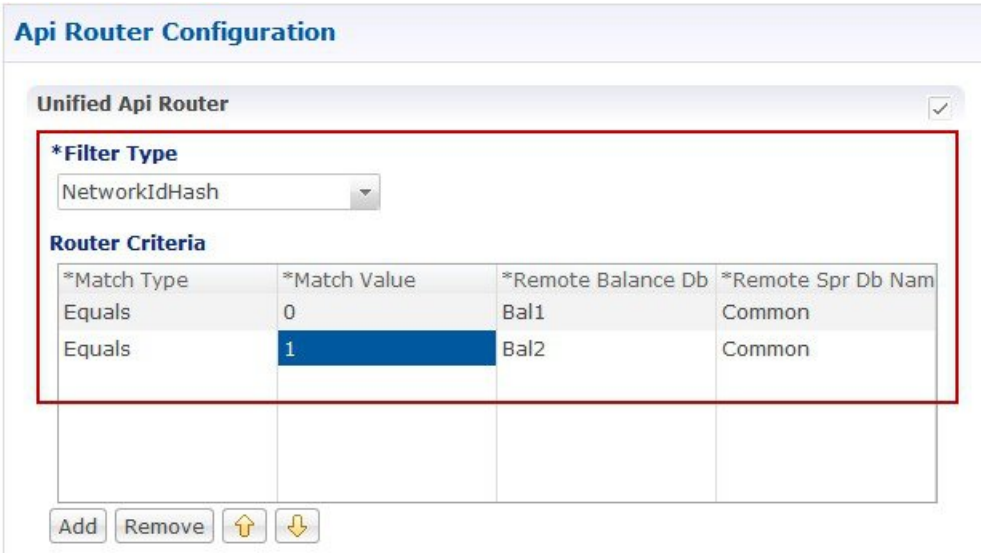
Balance Configuration

Figure 34: Balance Configuration



API Router Configuration

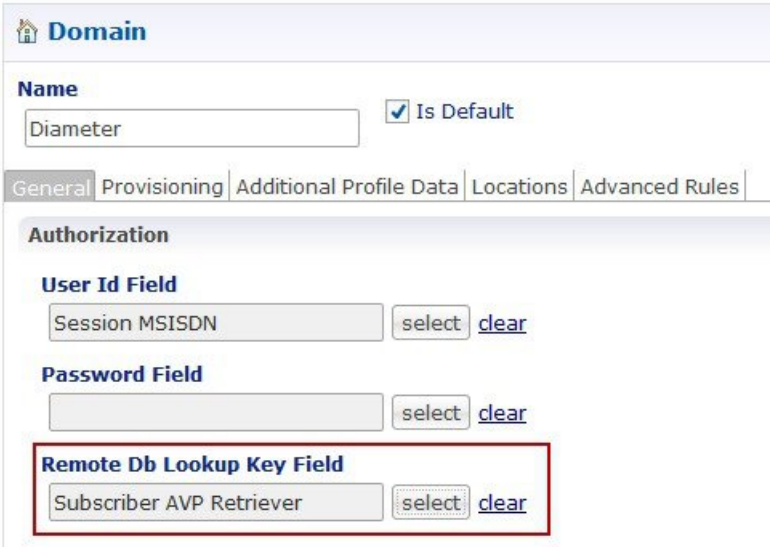
Figure 35: API Router Configuration



Common SPR Database and Multiple Balance Database based on SPR AVP

Domain Configuration

Figure 36: Domain Configuration

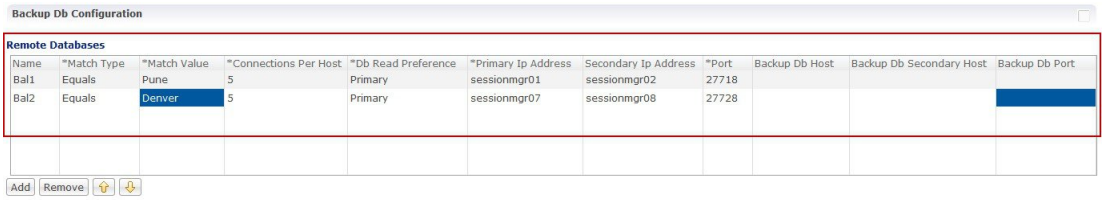


USuM Configuration

Figure 37: USuM Configuration

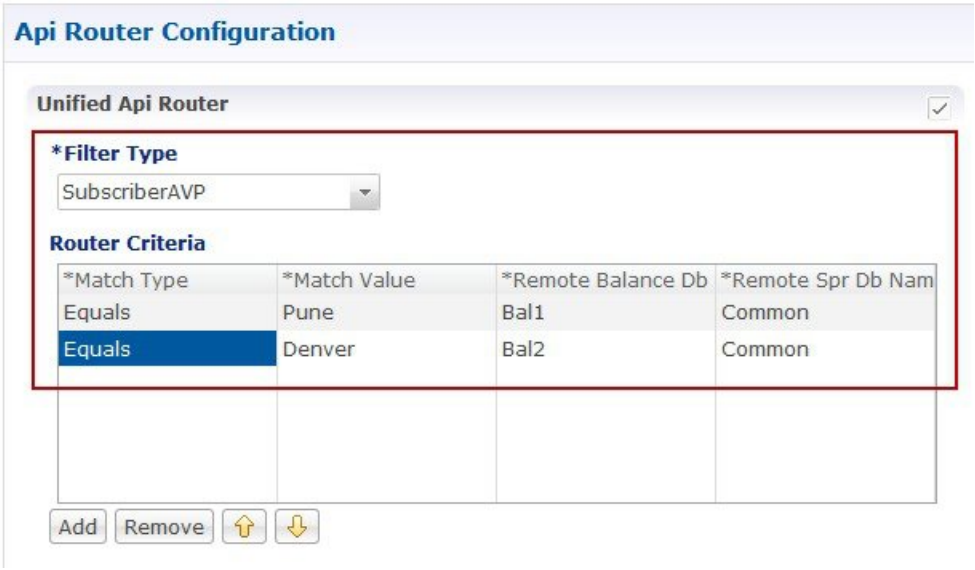
Balance Configuration

Figure 38: Balance Configuration



API Router Configuration

Figure 39: API Router Configuration



Rebalance

For hash-based balance and common SPR, rebalance is supported. It means old balance data can be rebalanced to new balance databases without need of re-provisioning.

Rebalance can be done by executing the following OSGi commands:

- `rebalanceByHash` — Rebalance with same balance shards (shards here means internal balance shards).
- `rebalanceByHash` *[oldShardCount] [newShardCount]* — Rebalance to change (increase/decrease) balance shards.

Rebalance with same balance shard

This is applicable only for hash based balance databases. To add new database to existing database, perform the following steps:

Step 1 Log in to Control Center and note down few subscriber which has balance.

Step 2 Change Policy Builder configuration: API Router, Balance, Domain, so on and publish the modified configuration.

Step 3 Add parameter `maxHash` in `qns.conf` file.

a) Value depends on number of databases.

For example, if there are two balance databases configured in Policy Builder, set value to 2.

```
-DmaxHash=2
```

Step 4 Add context path parameter `ua.context.path` and `apirouterContextPath` in `qns.conf` file. This is needed for Control Center to call via API router.

```
-DapirouterContextPath=/ua/soap

-Dua.context.path=/ua/backEnd
```

Step 5 Execute `copytoall.sh` and restart Policy Server (QNS) processes.

Step 6 Login to OSGi console on qns01.

```
telnet qns01 9091
```

Step 7 Execute `rebalanceByHash` command.

```
rebalanceByHash
```

Step 8 Log in to Control Center and verify subscriber still has balance noted in Step 1, on page 111.

Rebalance to Change Number of Balance Shards

This is applicable only for hash-based balance databases.

To increase the number of balance shards, perform the following steps:

1. Login to Control Center and note down *com.cisco.balance.dbs* few subscribers who have balance.
2. In the qns.conf file, add or edit *com.cisco.balance.dbs*.
   1. Value will be new shard number.

      Example — If you are increasing balance shards from 4 to 8, value should be set to 8.

      ```
      -Dcom.cisco.balance.dbs=8
      ```
3. Run `copytoall.sh` and restart qns processes.
4. Login to OSGi console on qns01.

   ```
   telnet qns01 9091
   ```
5. Run `rebalanceByHash` command.

   ```
   rebalanceByHash <old shard number> <new shard number>
   ```

   Example — If you are increasing balance shards from 4 to 8, old shard number is 4 and new shard number is 8.

   ```
   rebalanceByHash 4 8
   ```
6. Login to Control Center and verify subscriber still has balance noted in Step 1.

To decrease the number of balance shards, perform the following steps:

1. Login to Control Center and note down few subscribers who have balance.
2. Login to OSGi console on qns01.

   ```
   telnet qns01 9091
   ```
3. Run `rebalanceByHash` command.

   ```
   rebalanceByHash <old shard number> <new shard number>
   ```

   Example — If you are decreasing balance shards from 6 to 4, old shard number is 6 and new shard number is 4.

```
rebalanceByHash 6 4
```

4. In the qns.conf file, add or edit *com.cisco.balance.dbs*.
   1. Value will be new shard number

      Example — If you are decreasing balance shards from 6 to 4, value should be set to 4.

      ```
      -Dcom.cisco.balance.dbs=4
      ```

5. Run `copytoall.sh` and restart qns processes.
6. Login to Control Center and verify subscriber still has balance noted in Step1.

Configurations to Handle Traffic Switchover

When Policy Server (QNS) is Down

The following configurations are needed to enable `qns_hb.sh` script. This script stops Policy Server (QNS) processes from lb01/lb02 when all Policy Server (QNS) are down (that is, qns01,02..n).

Note To understand traffic switchover, refer to Load Balancer VIP Outage, on page 148.

Step 1 To enable script, add the following configuration in `/var/qps/config/deploy/csv/Configuration.csv` file:

```
mon_qns_lb,true,
```

For more information on how to add the parameter in `Configuration.csv` file, refer to *CPS Installation Guide for VMware* for 9.1.0 and later releases.

Note Any database that is not replicated across sites should not be configured for monitoring by `mon_db_for_callmodel.sh` script.

Step 2 To disable script, add the following configuration in `/var/qps/config/deploy/csv/Configuration.csv` file or remove `mon_qns_lb` tag from this CSV file:

```
mon_qns_lb,,
```

Step 3 Import CSV to JSON by executing the following command:

```
/var/qps/install/current/scripts/import/import_deploy.sh
```

Step 4 Execute the following command to validate the imported data:

```
cd /var/qps/install/current/scripts/deployer/support/

python jvalidate.py
```

Note The above script validates the parameters in the Excel/csv file against the ESX servers to make sure ESX server can support the configuration and deploy VMs.

Step 5 Reinitiate lb01 and lb02 by executing the following command:

```
/etc/init.d/vm-init
```

For more information on configuring GR features, refer to *CPS Mobile Configuration Guide*.

When Replicated (inter-site) Database is not Primary on a Site

Note To understand traffic switchover, refer to Load Balancer VIP Outage, on page 148.

Step 1 Add the list of databases that needs to be monitored in `mon_db_for_callmodel.conf` file (`/etc/broadhop/mon_db_for_callmodel.conf`) in Cluster Manager.

Important Contact your Cisco Technical Representative for more details.

Add the following content in the configuration file (`mon_db_for_callmodel.conf`):

Note The following is an example and needs to be changed based on your requirement:

```
#this file contains set names that are available in mongoConfig.cfg. Add set names one below other.
#Refer to README in the scripts folder.
SESSION-SET1
SESSION-SET2
BALANCE-SET1
SPR-SET1
```

Step 2 To enable switch-off of UAPI traffic when replicated (inter-site) configured databases are not primary on this site, add `STOP_UAPI=1` in `/etc/broadhop/mon_db_for_callmodel.conf` file.

Note To disable switch-off of UAPI traffic when replicated (inter-site) configured databases are not primary on this site, add `STOP_UAPI=0` (if this parameter is not defined,) in `/etc/broadhop/mon_db_for_callmodel.conf` file.

When we recover GR site, we have to manually start UAPI interface (if it is disabled) by executing the following command as a root user on lb01 and lb02:

```
echo "enable frontend https-api" | socat stdio /tmp/haproxy
```

Step 3 To configure the percentage value for session replica sets to be monitored from the configured session replica set list, set the `PERCENTAGE_SESS_DB_FAILURE` parameter in the `/etc/broadhop/mon_db_for_callmodel.conf` configuration file.

Use an integer from **`1`** to **`100`**.

Note `PERCENTAGE_SESS_DB_FAILURE` parameter should be configured such that it results in integer value and not float for the number of sets user expects to be down to trigger the fail over. For example, if there are 4 session replica-sets and user wants failover to be triggered only when two or more replica-sets are down (>= 2 condition), user should configure this value as 50 (%). User at times wrongly interprets this parameter as "more than given number of replica-sets". For example in above case, user intends to have a failover if more than one replica-set (so, this is equivalent to "> 1" condition) are down and configures a value which is more than 25 (for example, 45) which does not work. The expected value should meet the condition of ">= desired number of replica-sets to be down for failover trigger".

Step 4 Rebuild etc directory on cluster by executing the following command:

```
/var/qps/install/current/scripts/build/build_etc.sh
```

When Virtual IP (VIP) is Down

You can configure CPS to monitor VIPs. If any of the configured VIPs goes down, the configured databases from the local site are made secondary. However, if the databases at a site go down, the VIP is not shut down.

For VMware

Specify the configuration in `/etc/broadhop/mon_db_for_lb_failover.conf`.

Note The `/etc/broadhop/mon_db_for_lb_failover.conf` file contains the configuration for VIPs along with the database set names.

For OpenStack

- During fresh install, apply the configuration using `http://<cluman-ip>:8458/api/system/config/action/`.
- Once system is deployed, use PATCH API `http://<cluman-ip>:8458/api/system/config/application-config` to apply the configuration that enables monitoring of VIPs.

The new configuration `vipMonitorForLb` with the following format is created under applicationConfig:

```
vipMonitorForLb:
 vipName:
 - lbvip01
 - lbvip02
```

Where, vipName is the array of VIPs to be monitored.

In case of any issues, check the API log file `/var/log/orchestration-api-server.log` and the `/var/log/broadhop/scripts` directory (after system configuration) for any errors.

Configuring Session Database Percentage Failure

You can configure the percentage value of session replica sets to be monitored from the configured session replica set list.

For VMware

Specify the configuration in `/etc/broadhop/mon_db_for_callmodel.conf` by modifying the `PERCENTAGE_SESS_DB_FAILURE` parameter. Use an integer from **`1`** to **`100`**.

For example, PERCENTAGE_SESS_DB_FAILURE=50

Note `PERCENTAGE_SESS_DB_FAILURE` parameter should be configured such that it results in integer value and not float for the number of sets user expects to be down to trigger the fail over. For example, if there are 4 session replica-sets and user wants failover to be triggered only when two or more replica-sets are down (>= 2 condition), user should configure this value as 50 (%). User at times wrongly interprets this parameter as "more than given number of replica-sets". For example in above case, user intends to have a failover if more than one replica-set (so, this is equivalent to "> 1" condition) are down and configures a value which is more than 25 (for example, 45) which does not work. The expected value should meet the condition of ">= desired number of replica-sets to be down for failover trigger".

For OpenStack

- During fresh install, apply the configuration using `http://<cluman-ip>:8458/api/system/config/action`.
- Once system is deployed, use PATCH API `http://<cluman-ip>:8458/api/system/config/application-config` to apply the configuration.

The new configuration `dbMonitorForQns` with the following format is created under applicationConfig:

```
dbMonitorForQns:
stopUapi: "false"
percentageSessDBFailure: 50
setName:
- SESSION-SET1
- SESSION-SET2
- SESSION-SET3
- SESSION-SET4
```

In case of any issues, check the API log file `/var/log/broadhop/scripts/mon_db_for_callmodel_$DATE.log`.

Remote Databases Tuning Parameters

If remote databases are configured for balance or SPR, respective mongo connection parameters are required to be added in `/etc/broadhop/qns.conf` file.

Remote SPR

```
-DdbSocketTimeout.remoteSpr=1200
-DdbConnectTimeout.remoteSpr=600
-Dmongo.connections.per.host.remoteSpr=10
-Dmongo.threads.allowed.to.wait.for.connection.remoteSpr=10
-Dmongo.client.thread.maxWaitTime.remoteSpr=1200
```

Remote Balance

```
-DdbSocketTimeout.remoteBalance=1200
-DdbConnectTimeout.remoteBalance=600
-Dmongo.connections.per.host.remoteBalance=10
```

```
-Dmongo.threads.allowed.to.wait.for.connection.remoteBalance=10
-Dmongo.client.thread.maxWaitTime.remoteBalance=1200
```

SPR Query from Standby Restricted to Local Site only (Geo Aware Query)

Step 1 Add new entry for Site1 and Site 2 in `/etc/broadhop/pcrf/qns.conf` file on pcrfclient01.

```
-DsprLocalGeoSiteTag=Site1 ====> in Site1 qns.conf file

-DsprLocalGeoSiteTag=Site2 ====> in Site2 qns.conf file
```

Step 2 Execute `syncconfig.sh`. To reflect the above change, CPS needs to be restarted.

Step 3 Add tag to SPR MongoDBs.

a) Run `diagnostics.sh` command on pcrfclient01 and find SPR database primary member and port number.

```
diagnostics.sh --get_replica_status
```

b) Login to SPR database using the primary replica set hostname and port number.

For example, `mongo --host sessionmgr01 --port 27720`

c) Get the replica members by executing the following command:

Execute `rs.conf()` from any one member.

SAMPLE OUTPUT

```
set04:PRIMARY> rs.conf();
{
        "_id" : "set04",
        "version" : 319396,
        "members" : [
                {
                        "_id" : 0,
                        "host" : "pcrfclient-arbiter-site3:37720",
                        "arbiterOnly" : true
                },
                {
                        "_id" : 1,
                        "host" : "sessionmgr01-site1:27720",
                        "priority" : 2
                },
                {
                        "_id" : 2,
                        "host" : "sessionmgr02-site1:27720",
                        "priority" : 2
                },
                {
                        "_id" : 3,
                        "host" : "sessionmgr05-site1:27720",
                        "priority" : 2
                },
                {
                        "_id" : 4,
                        "host" : "sessionmgr06-site1:27720",
                        "votes" : 0,
                        "priority" : 2
```

```
                },
                {
                        "_id" : 5,
                        "host" : "sessionmgr01-site2:27720"
                },
                {
                        "_id" : 6,
                        "host" : "sessionmgr02-site2:27720"
                },
                {
                        "_id" : 7,
                        "host" : "sessionmgr05-site2:27720"
                },
                {
                        "_id" : 8,
                        "host" : "sessionmgr06-site2:27720",
                        "votes" : 0
                }
        ],
        "settings" : {
                "heartbeatTimeoutSecs" : 1
        }
}
```

d) Now from the list, find out the members of Site1 and Site2 to be tagged (excluding the arbiter). After finding member, execute the following command from Primary member to tag members.

Note To modify priorities, you must update the `members` array in the replica configuration object. The array index begins with 0. The array index value is different than the value of the replica set member's `members[n]._id` field in the array.

```
conf = rs.conf();
conf.members[1].tags = { "sprLocalGeoSiteTag": "Site1" }
conf.members[2].tags = { "sprLocalGeoSiteTag": "Site1"}
conf.members[3].tags = { "sprLocalGeoSiteTag": "Site1"}
conf.members[4].tags = { "sprLocalGeoSiteTag": "Site1"}
conf.members[5].tags = { "sprLocalGeoSiteTag": "Site2"}
conf.members[6].tags = { "sprLocalGeoSiteTag": "Site2"}
conf.members[7].tags = { "sprLocalGeoSiteTag": "Site2"}
conf.members[8].tags = { "sprLocalGeoSiteTag": "Site2"}
rs.reconfig(conf);
```

Note This is a sample output. Configuration, members and tag can be different as per your environment.

`conf.members[1]` means member with *_id = 1* in output of `rs.conf()`.

After executing this command, primary member can be changed.

Verify tags are properly set by log in on any member and executing the following command:

```
rs.conf();
```

SAMPLE OUTPUT

```
set04:PRIMARY> rs.conf();
{
        "_id" : "set04",
        "version" : 319396,
        "members" : [
                {
                        "_id" : 0,
                        "host" : "pcrfclient-arbiter-site3:37720",
```

```
        "arbiterOnly" : true
},
{
        "_id" : 1,
        "host" : "sessionmgr01-site1:27720",
        "priority" : 2,
        "tags" : {
                "sprLocalGeoSiteTag" : "Site1"
        }
},
{
        "_id" : 2,
        "host" : "sessionmgr02-site1:27720",
        "priority" : 2,
        "tags" : {
                "sprLocalGeoSiteTag" : "Site1"
        }
},
{
        "_id" : 3,
        "host" : "sessionmgr05-site1:27720",
        "priority" : 2,
        "tags" : {
                "sprLocalGeoSiteTag" : "Site1"
        }
},
{
        "_id" : 4,
        "host" : "sessionmgr06-site1:27720",
        "votes" : 0,
        "priority" : 2,
        "tags" : {
                "sprLocalGeoSiteTag" : "Site1"
        }
},
{
        "_id" : 5,
        "host" : "sessionmgr01-site2:27720",
        "tags" : {
                "sprLocalGeoSiteTag" : "Site2"
        }
},
{
        "_id" : 6,
        "host" : "sessionmgr02-site2:27720",
        "tags" : {
                "sprLocalGeoSiteTag" : "Site2"
        }
},
{
        "_id" : 7,
        "host" : "sessionmgr05-site2:27720",
        "tags" : {
                "sprLocalGeoSiteTag" : "Site2"
        }
},
{
        "_id" : 8,
        "host" : "sessionmgr06-site2:27720",
        "votes" : 0,
        "tags" : {
                "sprLocalGeoSiteTag" : "Site2"
        }
}
```

```
        ],
        "settings" : {
                "heartbeatTimeoutSecs" : 1
        }
}
```

Step 4 Repeat Step 3, on page 117 for all other sites. Tag names should be unique for each site.

This change overrides the read preference configured in **USuM Configuration** in Policy Builder.

Step 5 Execute `rs.reconfig()` command to make the changes persistent across replica-sets.

Balance Location Identification based on End Point/Listen Port

Prerequisites

Policy Builder configuration on both the sites should be the same.

Configuration

Consider there are two sites: Site1 (Primary) and Site2 (Secondary).

Add new entry on Site2 in `haproxy.cfg` (`/etc/haproxy/haproxy.cfg`) file listening on port 8081 (or any other free port can be used) with custom header *RemoteBalanceDbName*. Same configuration needs to be done on both load balancers.

```
listen pcrf_a_proxy lbvip01:8081
mode http
reqadd RemoteBalanceDbName:\ BAL_SITE1
balance roundrobin
option httpclose
option abortonclose
option httpchk GET /ua/soap/KeepAlive
server qns01_A qns01:8080 check inter 30s
server qns02_A qns02:8080 check inter 30s
server qns03_A qns03:8080 check inter 30s
server qns04_A qns04:8080 check inter 30s
# If there are more qns add all entries here
```

where,

RemoteBalanceDbName is the custom header.

BAL_SITE1 is the remote database name configured in Remote Balance Lookup based on IMSI/MSISDN Prefix, on page 98.

Figure 40: Balance Site

Balance Query Restricted to Local Site

The following steps need to be performed for balance query from restricted to local site only (Geo aware query) during the database failover:

Consider there are two sites: Site1 and Site2

Note If there are more than one balance databases, follow the below mentioned steps for all the databases.

The following steps are not needed to be performed for backup or hot standby databases.

Step 1 Add new entry in Site1 `qns.conf` file (`/etc/broadhop/qns.conf`) on Cluster Manager.

```
-DbalanceLocalGeoSiteTag=Site1
```

a) Run `copytoall.sh` to restart the qns processes.

Step 2 Add new entry on Site2 `qns.conf` file (`/etc/broadhop/qns.conf`) on Cluster Manager.

```
-DbalanceLocalGeoSiteTag=Site2
```

a) Run `copytoall.sh` to restart the qns processes.

Step 3 Add balance geo tag in MongoDBs.

a) Run `diagnostics.sh` command on pcrfclient01 and find balance database primary member and port number.

```
$ diagnostics.sh --get_replica_status
```

b) Login to balance database using the primary replica set hostname and port number.

For example, `$ mongo --host sessionmgr01 --port 27720`

c) Get the balance database replica members information.

Execute `rs.conf()` from any one member.

SAMPLE OUTPUT

```
set04:PRIMARY> rs.conf();
{
        "_id" : "set04",
        "version" : 319396,
        "members" : [
                {
                        "_id" : 0,
```

```
                        "host" : "pcrfclient-arbiter-site3:27718",
                        "arbiterOnly" : true
                },
                {
                        "_id" : 1,
                        "host" : "sessionmgr01-site1:27718",
                        "priority" : 4
    },
                {
                        "_id" : 2,
                        "host" : "sessionmgr02-site1:27718",
                        "priority" : 3
                },
                {
                        "_id" : 3,
                        "host" : "sessionmgr01-site2:27718",
                        "priority" : 2
                },
                {
                        "_id" : 4,
                        "host" : "sessionmgr02-site2:27718"
                        "priority" : 1
                }
        ],
        "settings" : {
                "heartbeatTimeoutSecs" : 1
        }
}
```

d) Now from the list, find out the members of Site1 and Site2 to be tagged (excluding the arbiter).

e) After finding member, execute the following command from primary member to tag members.

```
conf = rs.conf();
conf.members[1].tags = { "balanceLocalGeoSiteTag": "Site1" }
conf.members[2].tags = { "balanceLocalGeoSiteTag": "Site1"}
conf.members[3].tags = { "balanceLocalGeoSiteTag": "Site2"}
conf.members[4].tags = { "balanceLocalGeoSiteTag": "Site2"}
rs.reconfig(conf);
```

Note This is a sample configuration. Members and tag can be different according to your deployment.

`conf.members[1]` means member with *_id = 1* in output of `rs.conf()`.

After tagging the members, primary member may get changed if all the members have same priority.

To verify that the tags are properly set, log in to any member and execute `rs.conf()` command.

SAMPLE OUTPUT

```
set04:PRIMARY> rs.conf();
{
        "_id" : "set04",
        "version" : 319396,
        "members" : [
                {
                        "_id" : 0,
                        "host" : "pcrfclient-arbiter-site3:27718",
                        "arbiterOnly" : true
                },
                {
                        "_id" : 1,
                        "host" : "sessionmgr01-site1:27718",
                        "priority" : 4,
                        "tags" : {
                                "balanceLocalGeoSiteTag" : "Site1"
```

```
                        }
                },
                {
                        "_id" : 2,
                        "host" : "sessionmgr02-site1:27718",
                        "priority" : 3,
                        "tags" : {
                                "balanceLocalGeoSiteTag" : "Site1"
                        }
                },
                {
                        "_id" : 3,
                        "host" : "sessionmgr01-site2:27718",
                        "priority" : 2,
                        "tags" : {
                                "balanceLocalGeoSiteTag" : "Site2"
                        }
                },
                {
                        "_id" : 4,
                        "host" : "sessionmgr01-site2:27718",
                        "priority" : 1,
                        "tags" : {
                                "balanceLocalGeoSiteTag" : "Site2"
                        }
                }
        ],
        "settings" : {
                "heartbeatTimeoutSecs" : 1
        }
}
```

Session Query Restricted to Local Site during Failover

The following steps need to be performed for session query from restricted to local site only (Geo aware query) during the database failover:

Consider there are two sites: Site1 and Site2

Note If there are more than one session databases, follow the below mentioned steps for all the databases.

The following steps are not needed to be performed for backup or hot standby databases.

This geo tagging is applicable only during **database failover time period**. In normal case, session database query/update always happen on primary member.

Step 1 Add new entry in Site1 `qns.conf` file (`/etc/broadhop/qns.conf`) on Cluster Manager.

```
-DsessionLocalGeoSiteTag=Site1
```

a) Run `copytoall.sh` to restart the qns processes.

Step 2 Add new entry on Site2 `qns.conf` file (`/etc/broadhop/qns.conf`) on Cluster Manager.

```
-DsessionLocalGeoSiteTag=Site2
```

a) Run `copytoall.sh` to restart the qns processes.

Step 3 Add session geo tag in MongoDBs.

a) First, get the session database replica members information.

Execute `rs.conf()` from any one member.

SAMPLE OUTPUT

```
set04:PRIMARY> rs.conf();
{
        "_id" : "set04",
        "version" : 319396,
        "members" : [
                {
                        "_id" : 0,
                        "host" : "pcrfclient-arbiter-site3:27717",
                        "arbiterOnly" : true
                },
                {
                        "_id" : 1,
                        "host" : "sessionmgr01-site1:27717",
                        "priority" : 4
   },
                {
                        "_id" : 2,
                        "host" : "sessionmgr02-site1:27717",
                        "priority" : 3
                },
                {
                        "_id" : 3,
                        "host" : "sessionmgr01-site2:27717",
                        "priority" : 2
                },
                {
                        "_id" : 4,
                        "host" : "sessionmgr02-site2:27717"
                        "priority" : 1
                }
        ],
        "settings" : {
                "heartbeatTimeoutSecs" : 1
        }
}
```

b) Now from the list, find out the members of Site1 and Site2 to be tagged (excluding the arbiter).

c) After finding member, execute the following command from Primary member to tag members.

Note To modify priorities, you must update the `members` array in the replica configuration object. The array index begins with 0. The array index value is different than the value of the replica set member's `members[n]._id` field in the array.

```
conf = rs.conf();
conf.members[1].tags = { "sessionLocalGeoSiteTag": "Site1" }
conf.members[2].tags = { "sessionLocalGeoSiteTag": "Site1"}
conf.members[3].tags = { "sessionLocalGeoSiteTag": "Site2"}
conf.members[4].tags = { "sessionLocalGeoSiteTag": "Site2"}
rs.reconfig(conf);
```

THIS IS SAMPLE CONFIG, MEMBERS and TAG can be different according to your deployment.

`conf.members[1]` means member with "_id" = "1" in output of rs.conf();

After executing this command, primary member may get changed if all members have same priority.

Verify that the tags are properly set by log in to on any member and executing `rs.conf()` command.

SAMPLE OUTPUT

```
set04:PRIMARY> rs.conf();
{
        "_id" : "set04",
        "version" : 319396,
        "members" : [
                {
                        "_id" : 0,
                        "host" : "pcrfclient-arbiter-site3:27717",
                        "arbiterOnly" : true
                },
                {
                        "_id" : 1,
                        "host" : "sessionmgr01-site1:27717",
                        "priority" : 4,
                        "tags" : {
                                "sessionLocalGeoSiteTag" : "Site1"
                        }
                },
                {
                        "_id" : 2,
                        "host" : "sessionmgr02-site1:27717",
                        "priority" : 3,
                        "tags" : {
                                "sessionLocalGeoSiteTag" : "Site1"
                        }
                },
                {
                        "_id" : 3,
                        "host" : "sessionmgr01-site2:27717",
                        "priority" : 2,
                        "tags" : {
                                "sessionLocalGeoSiteTag" : "Site2"
                        }
                },
                {
                        "_id" : 4,
                        "host" : "sessionmgr01-site2:27717",
                        "priority" : 1,
                        "tags" : {
                                "sessionLocalGeoSiteTag" : "Site2"
                        }
                }
        ],
        "settings" : {
                "heartbeatTimeoutSecs" : 1
        }
}
```

Publishing Configuration Changes When Primary Site becomes Unusable

Step 1 Configure auto SVN sync across sites on GR secondary site.

a) Configure this site's pcrfclient01 public IP in `/var/qps/config/deploy/csv/AdditionalHosts.csv` file of Cluster Manager (use same host name in 2.a, on page 126).

Example: `public-secondary-pcrfclient01,,XX.XX.XX.XX,`

For OpenStack, add the new hosts in the AdditionalHosts section. For more information, refer to */api/system/config/additional-hosts* section in *CPS Installation Guide for OpenStack*

b) Recreate SVN repository on secondary site pcrfclient01/02.

Note Take backup of SVN repository for rollback purpose.

From pcrfclient01, execute the following commands:

```
/bin/rm -fr /var/www/svn/repos

/usr/bin/svnadmin create /var/www/svn/repos

/etc/init.d/vm-init-client
```

From pcrfclient02, execute the following commands:

```
/bin/rm -fr /var/www/svn/repos

/usr/bin/svnadmin create /var/www/svn/repos

/etc/init.d/vm-init-client
```

c) Login to pcrfclient01 and recover svn using `/var/qps/bin/support/recover_svn_sync.sh` script.

d) Verify SVN status using `diagnostics.sh` script from Cluster Manager. Output should look like:

```
diagnostics.sh --svn

CPS Diagnostics HA Multi-Node Environment
--------------------------
Checking svn sync status between pcrfclient01 and pcrfclient02...[PASS]
```

Step 2 Configure auto SVN sync across site on GR primary site.

For OpenStack, add the new hosts in the AdditionalHosts section. For more information, refer to */api/system/config/additional-hosts* section in *CPS Installation Guide for OpenStack*

a) Configure remote/secondary pcrfclient01 public IP in `/var/qps/config/deploy/csv/AdditionalHosts.csv` file of Cluster Manager.

Example: `public-secondary-pcrfclient01,,XX.XX.XX.XX,`

b) Configure svn_slave_list as `svn_slave_list,pcrfclient02 public-secondary-pcrfclient01` in `/var/qps/config/deploy/csv/Configuration.csv` file of Cluster Manager (replace public-secondary-pcrfclient01 with the host name assigned in Step 2 a).

c) Configure svn recovery to be every 10 minutes that is, `auto_svn_recovery,enabled` in `/var/qps/config/deploy/csv/Configuration.csv` file of Cluster Manager.

d) Execute `/var/qps/install/current/scripts/import/import_deploy.sh` on Cluster Manager.
e) Execute the following command to validate the imported data:

```
cd /var/qps/install/current/scripts/deployer/support/

python jvalidate.py
```

Note The above script validates the parameters in the Excel/csv file against the ESX servers to make sure ESX server can support the configuration and deploy VMs.

f) Login to pcrfclient01 and re-initiate pcrfclient01 using `/etc/init.d/vm-init-client` command.
g) Verify SVN status using `diagnostics.sh` script from Cluster Manager. Output should look like (Wait for maximum 10 mins as per cron interval for PASS status):

```
diagnostics.sh --svn
CPS Diagnostics HA Multi-Node Environment
--------------------------
Checking svn sync status between pcrfclient01 & pcrfclient02...[PASS]
Checking svn sync status between pcrfclient01 & remote-pcrfclient01...[PASS]
```

Test scenario

1. Both primary and secondary sites are up:
   - Login to primary site Policy Builder and update policy.
   - Verify primary site SVNs are in sync between pcrfclient01 and 02.
   - Verify primary site and secondary site SVN are in sync (this takes approx 10 mins as per cron interval).
2. Both primary and secondary sites are up (This is not recommended).
   - Login to secondary site Policy Builder and update policy.
   - Verify secondary site SVNs are in sync between pcrfclient01 and 02.
   - Verify primary site and secondary site SVN are in sync (this takes approx 10 mins as per cron interval).
3. Primary site up and secondary site down.
   - Login to primary site Policy Builder and update policy.
   - Verify primary site SVNs are in sync between pcrfclient01 and 02.
   - Recover secondary site. Verify primary site and secondary site SVN are in sync (this takes approx10 mins as per cron interval).
4. Secondary site up and primary site down.
   - Login to secondary site Policy Builder and update policy.
   - Verify secondary site SVNs are sync between pcrfclient01 and 02.
   - Recover primary site. Verify primary site and secondary site SVN are in sync (this takes approx 10 mins as per cron interval).

Graceful Cluster Shutdown

Utility script `/var/qps/bin/support/mongo/migrate_primary.sh` is a part of the build and will be available in newly installed or upgraded setups.

This utility simply reads the cluster configuration file and `mongoConfig.cfg` file, and migrates Mongo Primary status from one cluster to another before doing an upgrade by setting priority '1' (or given in the command). After upgrade, utility script again migrates Mongo Primary status to original cluster by restoring the original priority.

Here is a help page of `migrate_primary.sh` (`/var/qps/bin/support/mongo/migrate_primary.sh -help`) utility:

```
/var/qps/bin/support/mongo/migrate_primary.sh [--options]
[--db-options] [--hosts-all|--hosts-files <host-file-name> --hosts <host list..> ]
/var/qps/bin/support/mongo/migrate_primary.sh --restore <restore-filename>

  --hosts      CPS Host list which are upgrading (like sessionmgr01, lb01, pcrfclient01)
  --hosts-file File name which contains CPS upgrading hosts
  --hosts-all  Upgrading all hosts from this cluster
  --restore    Restore priority back

DB Options - you can provide multiples DBs
  --all         All databases in the configuration
  --spr         All SPR databases in the configuration
  --session     All Session databases in the configuration
  --balance     All Balance databases in the configuration
  --admin       All Admin databases in the configuration
  --report      All Report databases in the configuration
  --portal      All Portal databases in the configuration
  --audit       All Aduit databases in the configuration

Options:
  --setpriority <priority-num> Set specific priority (default is 1)
  --noprompt                   Do not ask verfiication & set priority, without y/n prompt
  --prompt                     Prompt before setting priority (Default)
  --nonzeroprioritychk         Validate all upgrading members have non-zero priority
  --zeroprioritychk            Validate all upgrading members have zero priority
  --debug                      For debug messages (default is non-debug)
  --force                      Set priority if replica set is not healthy or member down
case
  --h [ --help ]               Display this help and exit

Description:
Reconfiguring Mongo DB priorities while doing an upgrade of a set of session managers,
the Mongo DBs that exist on that session manager need to be moved to priority 1
(provided priority) so that they will never be elected as the primary at time of upgrade.

Examples:
  /var/qps/bin/support/mongo/migrate_primary.sh --all --hosts sessionmgr01
  /var/qps/bin/support/mongo/migrate_primary.sh --session --hosts-all
  /var/qps/bin/support/mongo/migrate_primary.sh --noprompt --spr --hosts sessionmgr01
sessionmgr02
  /var/qps/bin/support/mongo/migrate_primary.sh --setpriority 1 --all --hosts sessionmgr01
```

Note When you execute `migrate_primary.sh --restore` command, ignore the following error message if encountered:

```
[WARN] Failed to set priority to <priority number> to <hostname> (<set number>), due to "errmsg" :
"replSetReconfig should only be run on PRIMARY, but my state is SECONDARY; use the "force" argument to override",
```

Active/Active Geo HA - Multi-Session Cache Port Support

CPS supports communication with multiple session cache databases and process the Gx and Rx messages in Active/Active Geo HA model.

The criteria to select which Session Cache for Gx Requests for a given session should be based on configurable criteria, for example, origin realm and/or host. Wildcards are also supported. For Rx requests, CPS uses secondaryKey lookup to load session.

When session cache database is not available, backup database is used.

By default, Geo HA feature is not installed and is not enabled. To install and enable the Geo HA, perform the following steps:

Note To configure Active/Active Geo HA using APIs, refer to *Active-Active Geo HA Support* section in *CPS Installation Guide for OpenStack*.

Install Geo HA

Step 1 Edit feature file on cluster manager: `/etc/broadhop/pcrf/features`

Step 2 Remove policy feature entry from feature file.

```
com.broadhop.policy.feature
```

Step 3 Add policy Geo HA feature entry in feature file.

```
com.broadhop.policy.geoha.feature
```

Step 4 Execute the following commands:

Note *${CPS version}* - Input the CPS version in the following commands.

```
/var/qps/install/${CPS version}/scripts/build/build_all.sh
```

```
/var/qps/install/${CPS version}/scripts/upgrade/reinit.sh
```

Example:

```
/var/qps/install/9.0.0/scripts/build/build_all.sh
```

```
/var/qps/install/9.0.0//scripts/upgrade/reinit.sh
```

Enable Geo HA

Step 1 Set GeoHA flag to `true` in `qns.conf` file to enable Geo HA feature.

```
-DisGeoHAEnabled=true
```

Step 2 Remove `-DclusterFailureDetectionMS` parameter from `/etc/broadhop/qns.conf` file.

Step 3 Verify other site lb IP addresses are present in `/etc/hosts` file.

If entries are missing, modify `AdditionalHost.csv` to add entries in `/etc/hosts` file. Remote load balancer should be accessible from load balancer by host name.

Configuration

Consider there are two sites Site1, Site2. Both are active sites and Site1 failovers to Site2.

Session database is replicated across site. Session database of the site can be selected based on realm or host or local information.

Step 1 Configure the lookup type in `qns.conf` file. Possible values can be realm/host/local.

For example, `-DgeoHASessionLookupType=realm`

Note When session lookup type is set to “local”, local session database is used for read/write session irrespective of site lookup configuration. For “local” session lookup type, site lookup configuration is not required. Even if it is configured, it is not used. However, user still needs to add site and shards.

For “local” session lookup type, add the entry for “diameter” under **Lookaside Key Prefixes** under Cluster configuration (if it is not already configured) in Policy Builder.

For local session capacity planning, refer to Local Session Affinity - Capacity Planning, on page 134.

Note In case there is an error in configuring `geoHASessionLookupType`, CPS behaves incorrectly and drop messages. The following logs come continuously if there is an error in the configuration:

```
GeoHA is enabled, unknown lookuptype is configured: <>. Possible values
are…
```

Step 2 Clean up the following data from the database if any data exists.

a) (Session database) Clean the sessions:

```
session_cache_ops.sh --remove
```

b) Run `diagnostics.sh` command on pcrfclient01 and find session database primary member and port number.

```
diagnostics.sh --get_replica_status
```

c) Login to session database using the primary replica set hostname and port number.

For example, `mongo --host sessionmgr01 --port 27720`

d) (Admin database) Remove shard entries from shards collection:

```
use sharding
db.shards.remove({});
db.buckets.drop(); ===> This collection is not used in GeoHA any more, so deleting.
```

e) (Admin database) Clear endpoints:

```
use queueing
db.endpoints.remove({});
use diameter
db.endpoints.remove({});
```

f) Exit the database:

```
exit
```

Step 3 Enable dictionary reload flag (Only for GR) in `/etc/broadhop/qns.conf` file.

```
-DenableReloadDictionary=true
```

Step 4 Update the following parameters in `/etc/broadhop/qns.conf` file as per site IDs.

Examples:

```
-DGeoSiteName=Site1
-DSiteId=Site1
-DRemoteSiteId=Site2
```

Step 5 Add Sites - configure/add all physical sites.

a) Login to qns OSGi console. All the following commands are to be executed from OSGi console.

```
telnet qns01 9091
```

b) Run addsite command for each primary (active) site with its secondary site ID.

```
addsite <SiteId> <SecondarySiteId>
```

where,

<*SiteId*> is the primary site ID.

<*SecondarySiteId*> is the secondary site ID.

Example: `addsite Site1 Site2`

This primary and secondary site IDs should be in sync with following entries in `/etc/broadhop/qns.conf` file.

Examples:

`-DGeoSiteName=Site1` ===> (this should be <*SiteId*> from the above `addsite` command

`-DSiteId=Site1` ===> (this should be <*SiteId*> from the above `addsite` command)

`-DRemoteSiteId=Site2` ===> (this should be <*SecondarySiteId*> from above `addsite` command)

c) Configure Site to realm or host mapping.

User needs to configure all the realms for all the interfaces (such as, Gx, Rx, and so on) here:

```
addsitelookup <SiteId> <LookupValue>
```

where,

<*SiteId*> is the primary site ID.

<*LookupValue*> is the realm/host value.

If `geoHASessionLookupType` is configured as realm in Step 1, on page 130.

Provide lookup value as realm as follows:

Example: If you have multiple Gx and Gy clients connected to CPS and the details for realms of clients are as follows:

```
Client-1: pcef-1-Gx.cisco.com
Client-2: pcef-1-Gy.cisco.com
Client-3: pcef-2-Gx.cisco.com
Client-4: pcef-2-Gy.cisco.com
```

For example, Client-1 and Client-2 are connected to Site1. Client-3 and Client-4 are connected to the Site2 then,

```
addsitelookup Site1 pcef-1-Gx.cisco.com
addsitelookup Site1 pcef-1-Gy.cisco.com
addsitelookup Site2 pcef-2-Gx.cisco.com
addsitelookup Site2 pcef-2-Gy.cisco.com
```

If `geoHASessionLookupType` is configured as host in Step 1, on page 130.

Provide lookup value as host as follows:

Example: If you have multiple Gx and Gy clients connected to CPS and the details for hostnames of clients are as follows:

```
Client-1: pcef-1-Gx
Client-2: pcef-1-Gy
Client-3: pcef-2-Gx
Client-4: pcef-2-Gy
```

For example, Client-1 and Client-2 are connected to Site1. Client-3 and Client-4 are are connected to the Site2 then,

```
addsitelookup Site1 pcef-1-Gx
addsitelookup Site1 pcef-1-Gy
addsitelookup Site2 pcef-2-Gx
addsitelookup Site2 pcef-2-Gy
```

Note The pattern matching is supported for site lookup mapping. In case the incoming host/realm does not match any of the values configured under LookupValues, request is dropped with the following exception in log:

```
GeoHASiteMappingNotFound - No realm/host to site mapping matched for:
<incoming value>
```

Note If you configure the same realm/host look up pattern for both sites, application picks only one site look up (first one in the list for same realms/hosts) to process the incoming requests. If the configured realms/host look-ups entries are same for both sites and you send calls to any site, the application will always try to use one site look up (first one in the list for same realms/hosts) to process the sessions.

So, Cisco recommends to configure identical realms/hosts names for the sites, and not configure same realms/hosts for the both sites.

Other commands:

- `listsitelookup`: To see all the configured site lookup mapping.
- `deletesitelookup <SiteId> <LookupValue>`: To delete specific site lookup mapping.

d) Add Shards: Configure/add shards for the site.

`addshard <Seed1>[,<Seed2>] <Port> <Index> <SiteId> [<BackupDb>]`

where,

<SiteId> is the primary site of the shard. This maps the shard with site.

[<BackupDb>] is an optional parameter.

Example: `addshard sessionmgr01,sessionmgr02 27717 1 Site1`

Note By default, there may not be any default shards added when Geo HA is enabled. So add the shards starting from index 1.

To configure hot standby feature, use `addshard` command with backup database parameter:

`addshard <Seed1>[,<Seed2>] <Port> <Index> <SiteId> [<BackupDb>]`

Examples:

```
addshard sessionmgr09,sessionmgr10 27717 1 Site1 backup
addshard sessionmgr09,sessionmgr10 27717 2 Site1 backup
```

e) Rebalance the shards by executing the following command:

`rebalance <SiteId>`

Example: `rebalance Site1`

f) Migrate the shards by executing the following command:

`migrate <SiteId>`

Example: `migrate Site1`

Step 6 (Optional) The following parameter should be updated in `/etc/broadhop/qns.conf` file for SP Wi-Fi based deployments:

```
-DisRadiusBasedGeoHAEnabled=true
```

Note For SP Wi-Fi based deployments, lookup value can be configured as NAS IP of ASR1K or ASR9K in Step 5, on page 131.

Note RADIUS-based policy control is no longer supported in CPS 14.0.0 and later releases as 3GPP Gx Diameter interface has become the industry-standard policy control interface.

GR Configuration with Session Replication Across Sites

Login to pcrfclient01/02 to create/update rings from Policy Server (QNS) OSGi console. Assuming there are two session cache replica-sets. By default, Ring-1 Set-1 get configured automatically and remaining rings need to be configured manually.

Configure cluster-1 (Ring-1)

```
osgi> setSkRingSet 1 1 <hostname_primary_site_sessionmgr01>:11211,
<hostname_primary_site_sessionmgr02>:11211
Ring updated
osgi> setSkRingSet 1 2 <hostname_primary_site_sessionmgr03>:11211,
<hostname_primary_site_sessionmgr04>:11211
Ring updated
```

Configure cluster-2 (Ring-2)

```
telnet qns01 9091
osgi> createSkRing 2
Successfully added ring with ID: 2
osgi> setSkRingSet 2 1 <hostname_secondary_site_sessionmgr01>:11211,
<hostname_secondary_site_sessionmgr02>:11211
osgi> setSkRingSet 2 2 <hostname_secondary_site_sessionmgr03>:11211,
<hostname_secondary_site_sessionmgr04>:11211
```

An example configuration is given below:

- Configure cluster-1 (Ring-1):

  ```
  osgi> setSkRingSet 1 1 L2-CA-PRI-sessionmgr01:11211, L2-CA-PRI-sessionmgr02:11211
  Ring updated
  osgi> setSkRingSet 1 2 L2-CA-PRI-sessionmgr03:11211, L2-CA-PRI-sessionmgr04:11211
  Ring updated
  ```

- Configure cluster-2 (Ring-2):

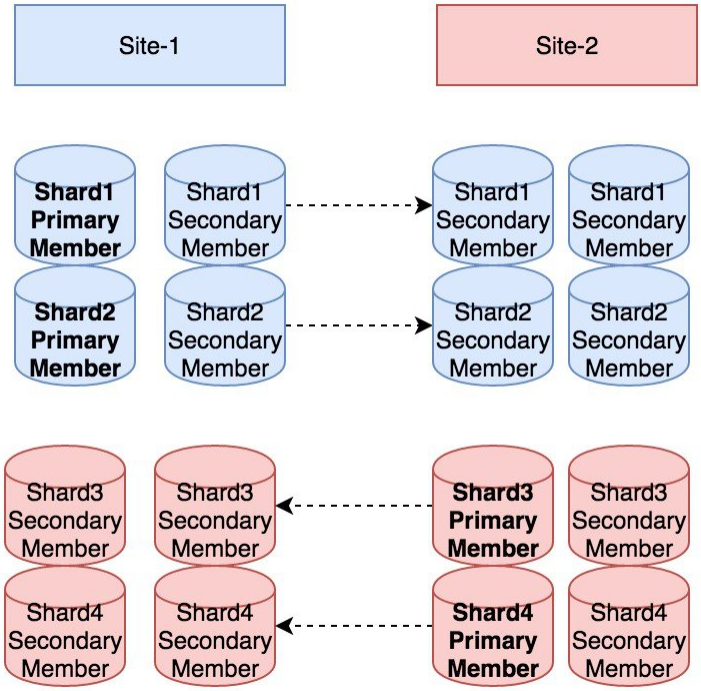
  ```
  telnet qns01 9091
  osgi> createSkRing 2
  Successfully added ring with ID: 2
  osgi> setSkRingSet 2 1 L2-CA-SEC-sessionmgr01:11211, L2-CA-SEC-sessionmgr02:11211
  osgi> setSkRingSet 2 2 L2-CA-SEC-sessionmgr03:11211, L2-CA-SEC-sessionmgr04:11211
  ```

Local Session Affinity - Capacity Planning

Consider there are two sites Site-1 and Site-2. Site-1 failovers to Site-2 and vice-versa.

With Local session affinity, both sites store new sessions to their local database in normal and failover condition. In normal conditions both sites need two session shards as shown in Figure 41: Site1 and Site2 - Normal Conditions, on page 135.

Figure 41: Site1 and Site2 - Normal Conditions

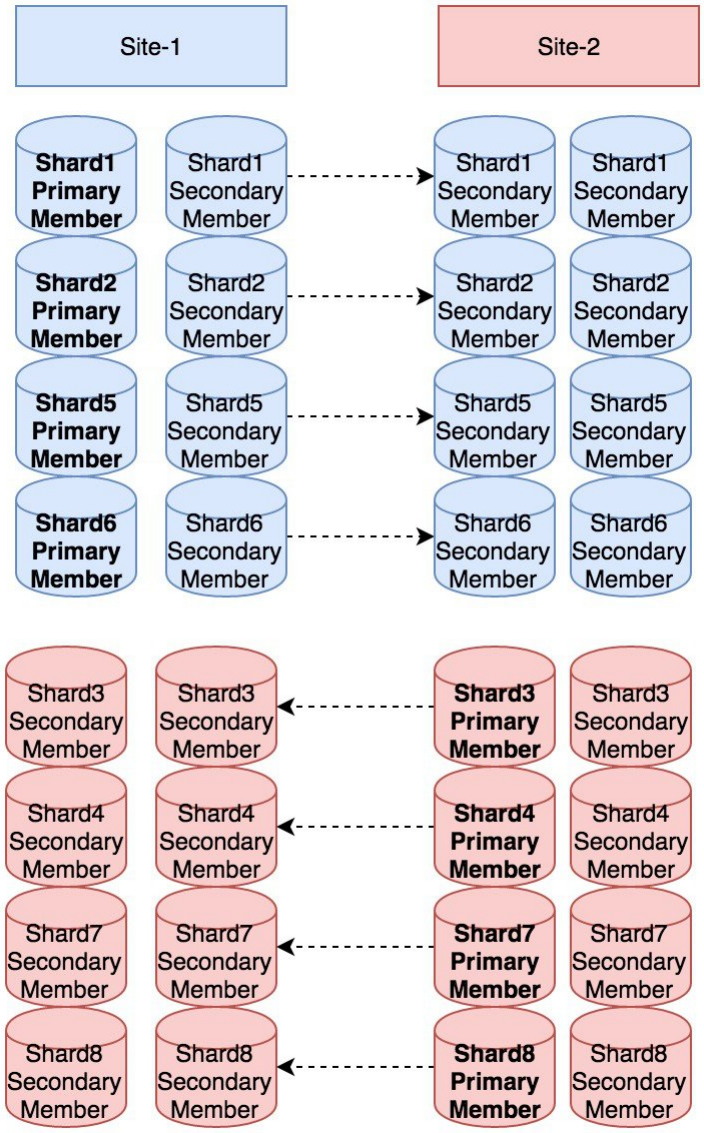


Site-1 writes sessions to Shard1 and Shard2, while Site-2 writes to Shard3 and Shard4.

Now consider, Site-2 goes down and Site-1 receives traffic for both the sites. Post failover, for new sessions, Site-1 will use Shard-1 and Shard-2 and it will not use the Shard-3 and Shard-4.

Site-1 needs to have extra session shards to handle the traffic from Site-1 and Site2 (after failover). Same for Site-2 (refer to Figure 42: Failover Scenario, on page 136 with additional shards).

Figure 42: Failover Scenario



Site-1 writes sessions to Shard1, Shard2, Shard5, Shard6 and Site-2 writes to Shard3, Shard4, Shard7, Shard8. Now when failover occurs both sites have additional local shards to accommodate sessions from another site.

Limitation

In this case, for profile refresh scenario, there is no 'smart' search (for example, IMSI-based match in given set of shards). In case a session for given profile is not found in concerned site's all session shards, search would be extended to all shards in all the sites.

For SP Wi-Fi deployments: Portal call flows are currently not supported in SP Wi-Fi Active-Active with session replication deployments. Currently, only ASR1K and ASR9K devices are supported for Active-Active deployments.

Note RADIUS-based policy control is no longer supported in CPS 14.0.0 and later releases as 3GPP Gx Diameter interface has become the industry-standard policy control interface.

Handling RAR Switching

When Gx session is created on Site2 and SPR update or Rx call comes on Site1, CPS sends Gx RAR from Site1 and in response PCEF sends RAA and next CCR request to Site1.

This leads to cross-site call switches from Site2 to Site1. If there are lot of call switches, Site1 may get overloaded.

By default, cross-site switching is enabled in CPS. To prevent cross-site switching, user needs to configure `-DRemoteGeoSiteName` parameter in `/etc/broadhop/qns.conf` file. This parameter enables cross-site communication for outbound messages like for RAR if we do not have DRA outside policy director (lb) and want to avoid RAR Switches.

Parameter Name: `-DRemoteGeoSiteName=<`*sitename*`>`

where, <*sitename*> is remote site name to be provided, and only to be added if we want to disable Gx-RAR switches from PCRF. It should match with `-DRemoteSiteId`.

Example:

```
-DRemoteSiteId=clusterA_SBY

-DRemoteGeoSiteName=clusterA_SBY
```

Prerequisite: Both remote site and local site policy server (QNS) should be able to communicate to load balancer on same interfaces. To change interface, flag `-Dcom.broadhop.q.if=<enter replication interface name>` can be used.

After configuring `-DRemoteGeoSiteName` parameter in `qns.conf` file, execute the following commands from Cluster Manager:

```
/var/qps/bin/control/copytoall.sh

/var/qps/bin/control/restartall.sh
```

If Redis IPC is used, make sure remote/peer policy director (lb) VM information is configured on local site for RAR switching to work. For more information, refer to Policy Director (lb) VM Information on Local Site, on page 33.

Configure Cross-site Broadcast Messaging

The cross-site broadcast message configuration is required when there are separate sessions (no replication for sessions DB) but common subscriber profile and subscriber provisioning event needs to be done on single site. In this case, the profile updates for subscriber sessions on remote sites need to be broadcasted to respective sites so that the corresponding RARs go from the remote sites to their respective diameter peers.

Edit `/etc/broadhop/qns.conf` file and add the following line:

```
-DclusterPeers=failover:(tcp://<remote-site-lb01>:<activemq port>,tcp://<remote-site-lb02>:<activemq port>)?updateURIsSupported=false!<remote-site-cluster-name>.default
```

where,

- <*remote-site-lb01*> is the IP address of the remote site lb01.
- <*activemq port*> is the port on which activemq is listening. Default is 61616.
- <*remote-site-lb02*> is the IP address of the remote site lb02.
- <*remote-site-cluster-name*> is the cluster name of remote site. To get the cluster name of remote site, check the parameter value of `-Dcom.broadhop.run.clusterId` in `/etc/broadhop/qns.conf` file on remote site.

Example:

```
-DclusterPeers=failover:(tcp://107.250.248.144:61616,tcp://107.250.248.145:61616)?updateURIsSupported=false!Cluster-Site-2.default
```

Example

The following example considers three sites (SITE-A, SITE-B and SITE-C) to configure cluster broadcast messaging between them.

Note Separator between two site configurations is colon (;).

- SITE-A configuration: Edit `/etc/broadhop/qns.conf` file and add the following lines:

```
-Dcom.broadhop.run.clusterId=Cluster-Site-A
-DclusterPeers=failover:(tcp://105.250.250.150:61616,tcp://105.250.250.151:61616)?updateURIsSupported=false!Cluster-SITE-B.default;failover:(tcp://105.250.250.160:61616,tcp://105.250.250.161:61616)?updateURIsSupported=false!Cluster-SITE-C.default
```

- SITE-B configuration: Edit `/etc/broadhop/qns.conf` file and add the following lines:

```
-Dcom.broadhop.run.clusterId=Cluster-Site-B
-DclusterPeers=failover:(tcp://105.250.250.140:61616,tcp://105.250.250.141:61616)?updateURIsSupported=false!Cluster-SITE-A.default;failover:(tcp://105.250.250.160:61616,tcp://105.250.250.161:61616)?updateURIsSupported=false!Cluster-SITE-C.default
```

- SITE-C configuration: Edit `/etc/broadhop/qns.conf` file and add the following lines:

```
-Dcom.broadhop.run.clusterId=Cluster-Site-C
-DclusterPeers=failover:(tcp://105.250.250.140:61616,tcp://105.250.250.141:61616)?updateURIsSupported=false!Cluster-SITE-A.default;failover:(tcp://105.250.250.150:61616,tcp://105.250.250.151:61616)?updateURIsSupported=false!Cluster-SITE-B.default
```

Configure Redundant Arbiter (arbitervip) between pcrfclient01 and pcrfclient02

After the upgrade is complete, if the user wants a redundant arbiter (ArbiterVIP) between pcrfclient01 and pcrfclient02, perform the following steps:

Currently, this is only supported for HA setups.

Step 1 Update the `AdditionalHosts.csv` and `VLANs.csv` files with the redundant arbiter information:

- **Update AdditionalHosts.csv:**

  Assign one internal IP for Virtual IP (arbitervip).

  Syntax:

  <alias for Virtual IP>,<alias for Virtual IP>,<IP for Virtual IP>

  For example,

  `arbitervip,arbitervip,`*< IP for Virtual IP>*

- **Update VLANs.csv:**

  Add a new column **Pcrfclient VIP Alias** in the `VLANs.csv` file to configure the redundant arbiter name for the pcrfclient VMs:

  Figure 43: VLANs.csv

1	VLAN Name	Network Target Name	Netmask	Gateway	VIP Alias	Pcrfclient VIP Alias	guestNic
2	Internal	VM Network	255.255.255.0	NA	lbvip02	arbitervip	eth0
3	Management	VLAN 94	255.255.255.0	NA	lbvip01		eth1
4	Gx	VM Network	255.255.255.0	NA	lbvip03		eth2
5							

  Execute the following command to import csv files into the Cluster Manager VM:

  ```
  /var/qps/install/current/scripts/import/import_deploy.sh
  ```

  This script converts the data to JSON format and outputs it to `/var/qps/config/deploy/json/`.

Step 2 SSH to the pcrfclient01 and pcrflclient02 VMs and run the following command to create arbitervip:

```
/etc/init.d/vm-init-client
```

Step 3 Synchronize `/etc/hosts` files across VMs by running the following command the Cluster Manager VM:

```
/var/qps/bin/update/synchosts.sh
```

Moving Arbiter from pcrfclient01 to Redundant Arbiter (arbitervip)

In this section we are considering the impacts to a session database replica set when the arbiter is moved from the pcrfclient01 VM to a redundant arbiter (arbitervip). The same steps need to be performed for SPR/balance/report/audit/admin databases.

Step 1 Remove pcrfclient01 from replica set (set01 is an example in this step) by executing the following command from Cluster Manager:

To find the replica set from where you want to remove pcrfclient01, refer to your `/etc/broadhop/mongoConfig.cfg` file.

```
build_set.sh --session --remove-members --setname set01
```

This command asks for member name and port number. You can find the port number from your `/etc/broadhop/mongoConfig.cfg` file.

```
Member:Port   --------> pcrfclient01:27717
pcrfclient01:27717
Do you really want to remove [yes(y)/no(n)]: y
```

Step 2 Verify whether the replica set member has been deleted by executing the following command from Cluster Manager:

```
diagnostics.sh --get_replica_status
```

```
|-------------------------------------------------------------------------------------------------|
| SESSION:set01                                                                                   |
|  Member-1  -  27717 : 221.168.1.5  -  PRIMARY   -  sessionmgr01 - ON-LINE  -  --------  -  1 |
|  Member-2  -  27717 : 221.168.1.6  -  SECONDARY -  sessionmgr02 - ON-LINE  -   0 sec    -  1 |
|-------------------------------------------------------------------------------------------------|
```

The output of `diagnostics.sh --get_replica_status` should not display pcrfclient01 as the member of replica set (set01 in this case).

Step 3 Change arbiter member from pcrfclient01 to redundant arbiter (arbitervip) in the `/etc/broadhop/mongoConfig.cfg` file by executing the following command from Cluster Manager:

```
vi /etc/broadhop/mongoConfig.cfg
[SESSION-SET1]
SETNAME=set01
OPLOG_SIZE=1024
ARBITER=pcrfclient01:27717             <-- change pcrfclient01 to arbitervip
ARBITER_DATA_PATH=/var/data/sessions.1
MEMBER1=sessionmgr01:27717
MEMBER2=sessionmgr02:27717
DATA_PATH=/var/data/sessions.1
[SESSION-SET1-END]
```

Step 4 Add a new replica set member by executing the following command from Cluster Manager:

```
build_set.sh --session --add-members --setname set01
              Replica-set Configuration
-------------------------------------------------------------------------
REPLICA_SET_NAME: set01
  Please select your choice for replica sets
             sharded (1) or non-sharded (2): 2
```

```
  Adding members to replica set             [  Done  ]
```

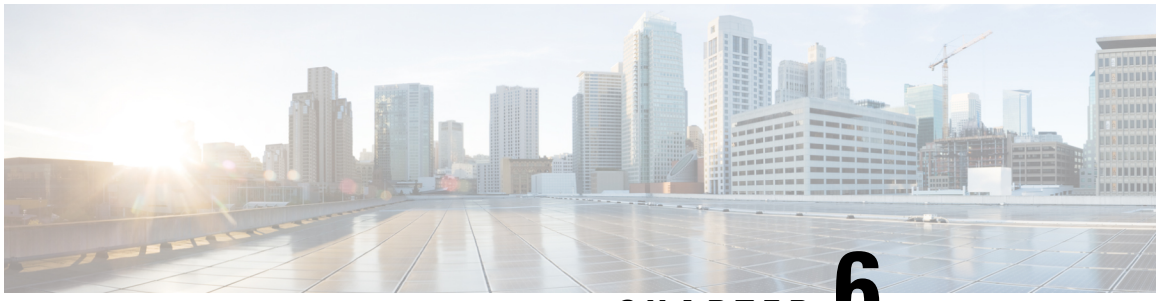
The progress of this script can be monitored in the `build_set` log file. For example, `/var/log/broadhop/scripts/build_set_23102017_220554.log`

Step 5 Verify whether the replica set member has been created by executing the following command from Cluster Manager:

```
diagnostics.sh --get_replica_status
```

```
|-------------------------------------------------------------------------------------------|
| SESSION:set01                                                                             |
|  Member-1  -  27717 : 221.168.1.5   -  PRIMARY    -  sessionmgr01  - ON-LINE  -  --------  -  1 |
|  Member-2  -  27717 : 221.168.1.6   -  SECONDARY  -  sessionmgr02  - ON-LINE  -   0 sec    -  1 |
|  Member-3  -  27717 : 221.168.1.9   -  ARBITER    -  arbitervip    - ON-LINE  -  --------  -  1 |
|-------------------------------------------------------------------------------------------|
```

The output of `diagnostics.sh --get_replica_status` should now display arbitervip as the member of replica set (set01 in this case).



CHAPTER 6

GR Failover Triggers and Scenarios

- Failover Triggers and Scenarios, on page 143

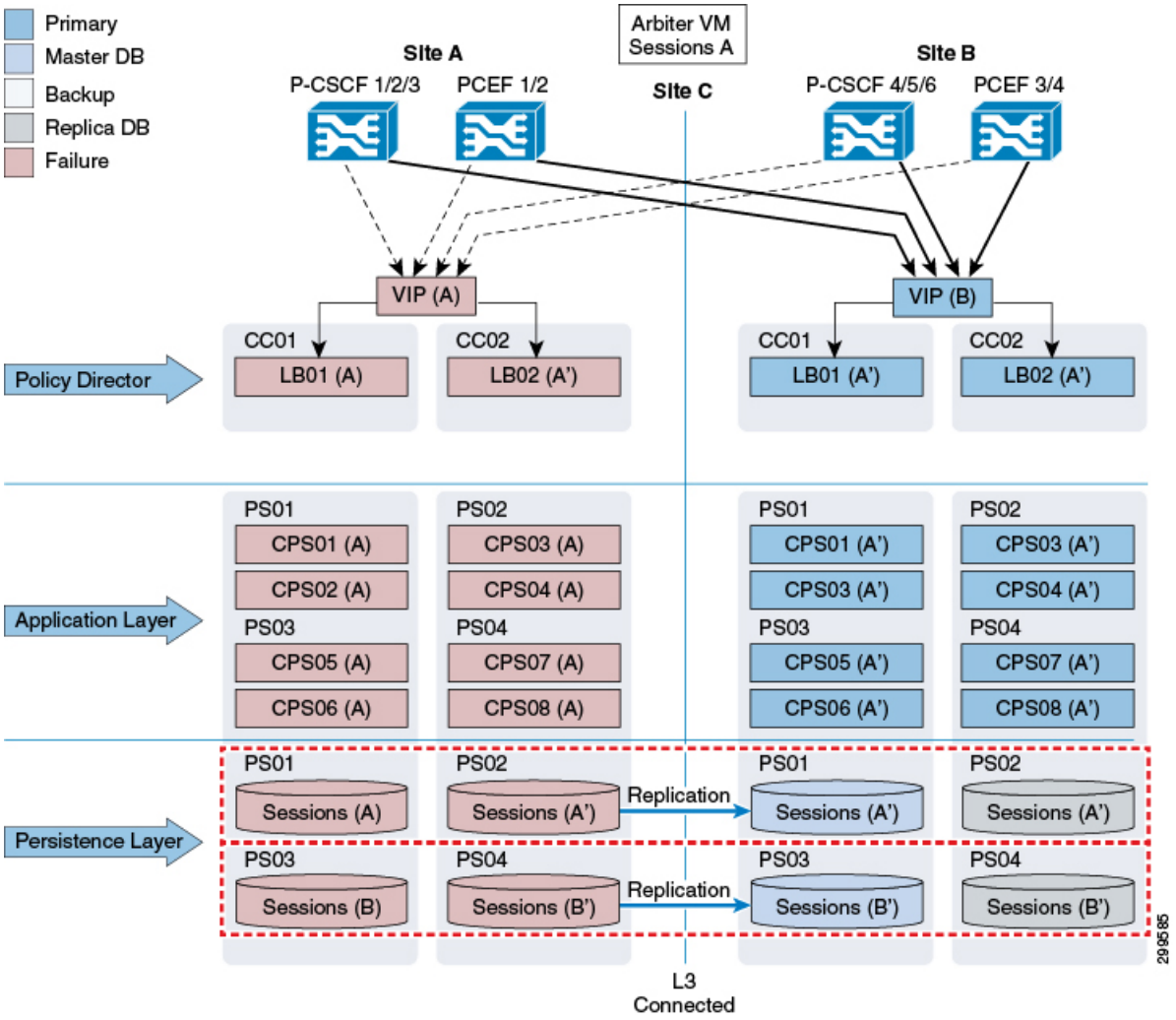
Failover Triggers and Scenarios

In Geographic Redundancy, there are multiple scenarios which could trigger a failover to another site.

Site Outage

As shown in the figure below, all P-GWs and P-CSCFs will direct traffic to the secondary site in the event of a complete outage of the primary site. Failover time will be dependent on failure detection timers on the P-GW and P-CSCF and the time it takes for the database replica set to elect a new Master database at the secondary site.

Figure 44: Outage of Primary Site



In order for Site A to be considered "ready for service" after an outage, all 3x tiers (Policy Director, Application Layer and Persistence Layer) must be operational.

At the Persistence (database replica set) level, MongoDB uses an operations log (oplog) to keep a rolling record of all operations that modify the data stored in the database. Any database operations applied on the Primary node are recorded on its oplog. Secondary members can then copy and apply those operations in an asynchronous process. All replica set members contain a copy of the oplog, which allows them to maintain the current state of the database. Any member can import oplog entries from any other member. Once the oplog is full, newer operations overwrite older ones.

When the replica members at Site A come back up after an outage and the connectivity between Sites A and B is restored, there are two possible recovery scenarios:

1. The oplog at Site B has enough history to fully resynchronize the whole replica set, for example the oplog did not get overwritten during the duration of the outage. In this scenario, the database instances at Site A will go into "Recovering" state once connectivity to Site B is restored. By default, when one of those instances catches up to within 10 seconds of the latest oplog entry of the current primary at Site B, the set will hold an election in order to allow the higher-priority node at Site A to become primary again.

2. The oplog at Site B does not have enough history to fully resynchronize the whole replica set (the duration of the outage was longer than what the system can support without overwriting data in the oplog). In this scenario, the database instances at Site A will go into "Startup2" state and stay in that state until we manually force a complete resynchronization (as they would be too stale to catch up with the current primary. A "too stale to catch up" message will appear in the mongodatabase.log or in the errmsg field when running rs.status(). For more information on manual resynchronization, Manual Recovery, on page 75.

   During a complete resynchronization, all the data is removed from the database instances at Site A and restored from Site B by cloning the Site B session database. All Read and Write operations will continue to use Site B during this operation.

Recovery time, holding time for auto recovery and so on depends upon TPS, latency, oplog size. For optimum values, contact your Cisco Technical Representative.

In CPS Release 7.5.0 and higher releases, at the Policy Director level, there is an automated mechanism to check availability of the Master database within the local site. When the Master database is not available, the policy director processes will be stopped and will not process with any incoming messages (Gx/Rx).

- This check runs at Site A (primary site).
- This check runs every 5 seconds (currently not configurable) and will determine whether the Master Sessions database is at Site A.

  It is possible to configure which databases the script will monitor (Sessions, SPR, Balance). By default, only the Sessions database is monitored.
- If the Master database is not available at Site A, the two Policy Director Processes (Loadatabasealancers) of site A will be stopped or remain stopped if recovering from a complete outage (as described in this section).
- In case of two replica sets, if one of the two Replica sets Master database is not available at Site A, the two Policy Director Processes (Loadatabasealancers) of site A will be stopped or remain stopped if recovering from a complete outage and the second replica set Master database will failover from Site A to Site B.

These above mentioned checks will prevent cross site communication for read/write operations. Once the site is recovered, P-GWs and P-CSCFs will start directing new sessions to Site A again.

For existing sessions, P-GWs will continue to send traffic to Site B until a message for the session (RAR) is received from Site A. That will happen, for example, when a new call is made and the Rx AAR for the new session is sent by the P-CSCF to Site A. Also, for existing Rx sessions, the P-CSCF will continue to send the traffic to Site B.

Gx Link Failure

As shown in the figure below, failure of the Gx link between a P-GW and the primary CPS node (Site A) results in the P-GW sending traffic to the secondary site (Site B). Failover time depends on failure detection timers on the P-GW.

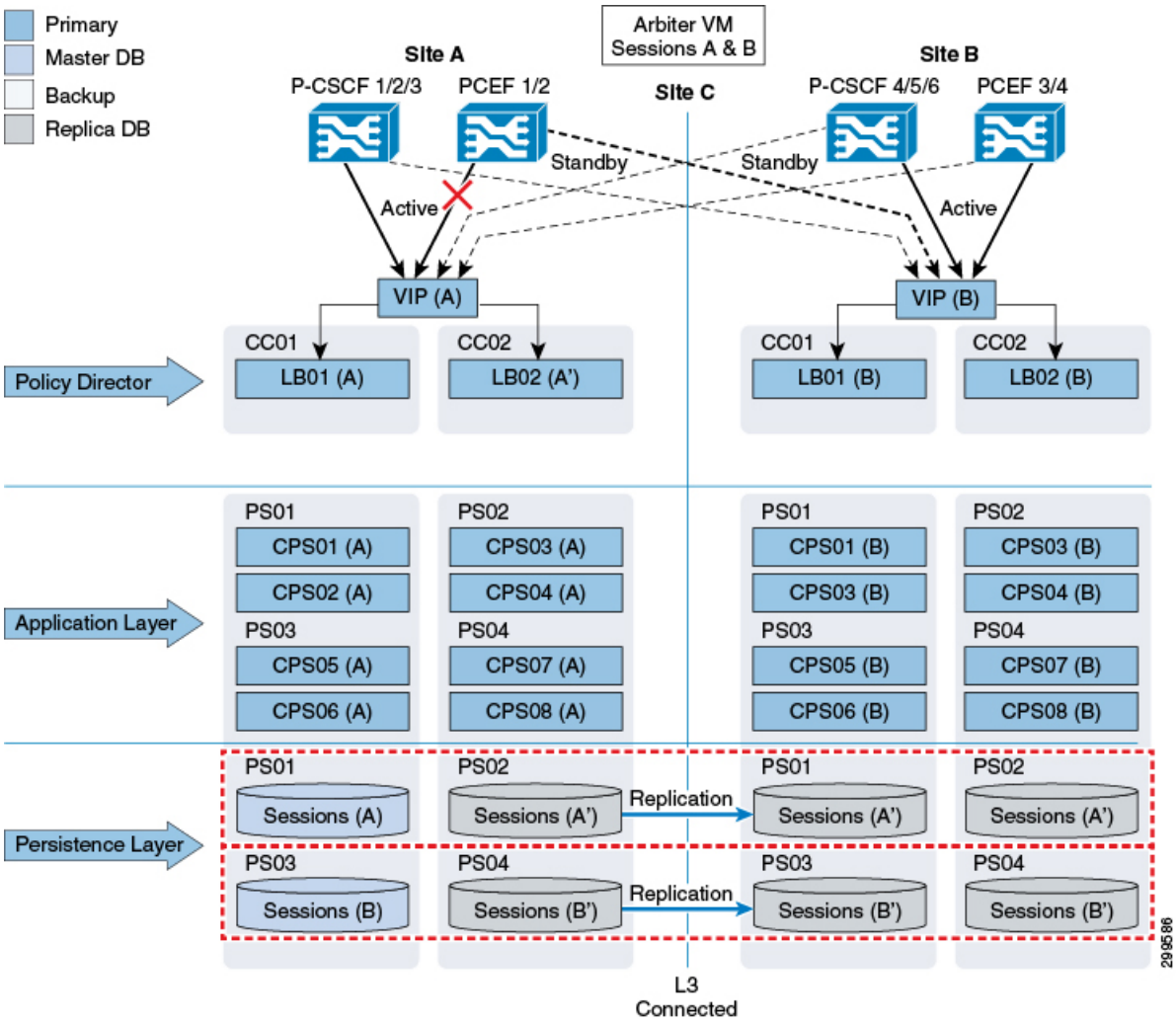
Gx transactions are processed at Site B.

If a session already exists, the CPS0x(B) VM handling the transaction at Site B retrieves the subscriber's session from the Master Sessions (A) database at Site A. New sessions as well as session updates are written across to the Master database at Site A.

Gx responses towards the P-GW (for example CCA), as well as Rx messages such as ASR that may be generated as a result of Gx transaction processing, is sent from Site B.

After receiving an Rx AAR at Site A, the resulting Gx RAR is proxied from the lb at Site A to the lb at Site B (as the P-GW is not reachable from Site A).

Figure 45: Gx Link Failure



Note For SP Wi-Fi deployments, if a link fails between PCEF 1/2 and CPS Site A, all messages coming from PCEF 1/2 to Site B are processed but messages generated from Site A for PCEF 1/2 are not proxied from Site B. P-CSCF communication is not applicable for SP Wi-Fi deployments.

Note RADIUS-based policy control is no longer supported in CPS 14.0.0 and later releases as 3GPP Gx Diameter interface has become the industry-standard policy control interface.

Failover Time Improvement

The subscriber impact for data calls (Gx only) has been reduced below 1 second for failures and recovery scenarios captured in following table:

Table 20: Failure Triggers Supported for Data (Gx) Call

Failover Trigger Scenarios	Operational Impact	Timeout Durations (ms)
VM shutdown on both primary and secondary members on local site while other replica-sets are running	Shutdown during Maintenance Window of multiple VM's on local site	800 ms or better
All replica-sets of local site are brought down	Power OFF of all session manager VM's	800 ms or better
VM startup of both failed local members in a replica-set	Startup or multiple VM's during Maintenance Window	800 ms or better
Replication or internal network link between two local members of a replica-set is down	None	800 ms or better

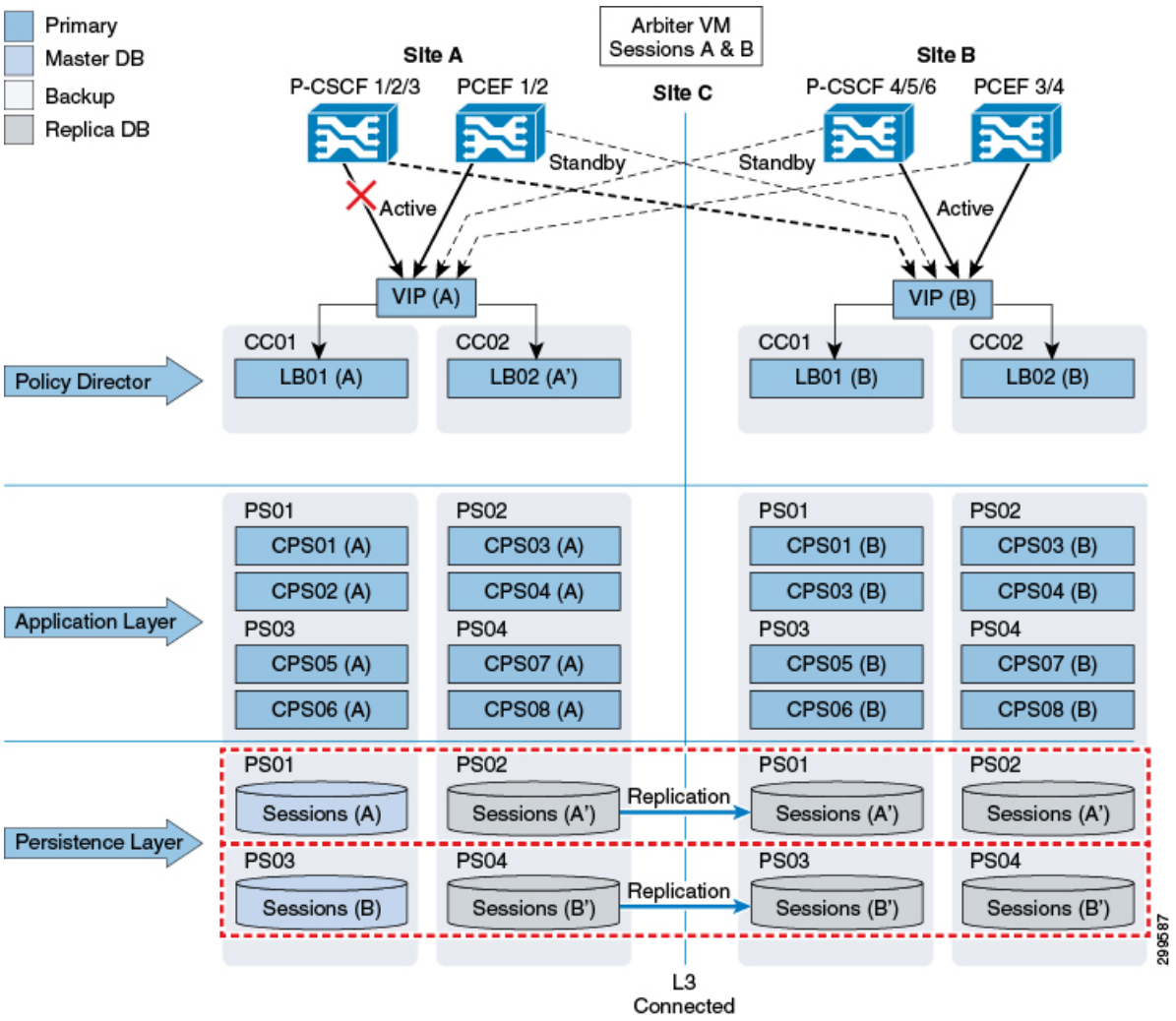
Rx Link Failure

As shown in the figure below, failure of the Rx link between a P-CSCF and the primary CPS node (Site A) results in the P-CSCF sending traffic to the secondary site (Site B). Failover time dependents on failure detection timers on the P-CSCF.

Rx transactions is processed at Site B. The CPS0x(B) VM handling the transaction at Site B attempts to do the binding by retrieving the Gx session from the Master Sessions(A) database at Site A. Session information is also written across to the Master database at Site A.

The Rx AAA back to the P-CSCF as well as the corresponding Gx RAR to the P-GW is sent from Site B.

Figure 46: Rx Link Failure



Note This link failure model does not apply for SP Wi-Fi depolyments.

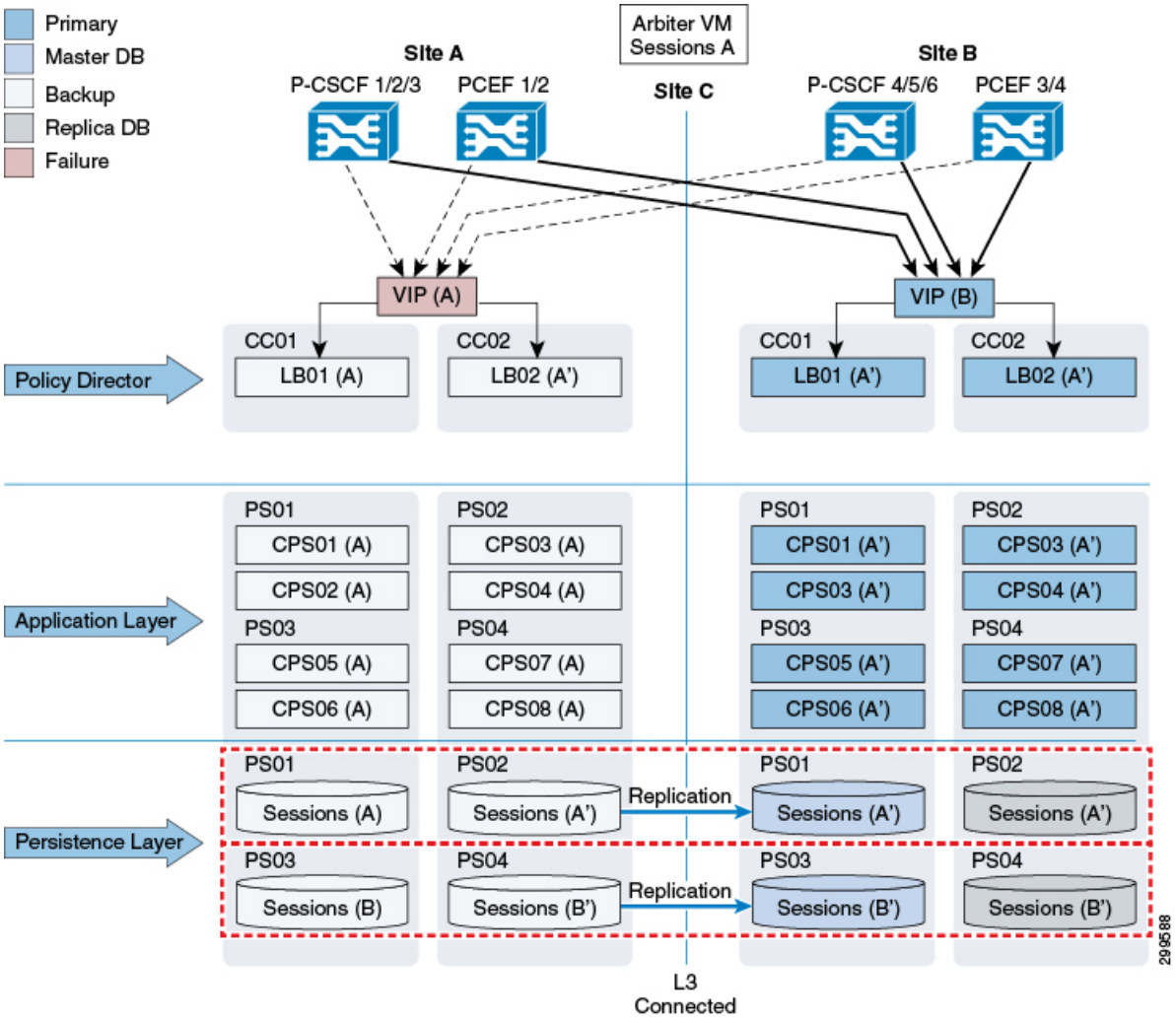
Load Balancer VIP Outage

As shown in the figure below, all P-GWs and P-CSCFs will direct traffic to the secondary site if both Load Balancer at the primary site is not available (which leads the VIP to be not available). Failover time will be dependent on failure detection timers on the P-GW and P-CSCF.

In order to avoid database writes from Site B to Site A, the system can be configured to monitor VIP availability and, if VIP is not available, lower the priority of the database instances at Site A to force the election of a new Master database at Site B.

By default, VIP availability is monitored every 60 seconds.

Figure 47: Load Balancer VIP Outage



Load Balancer/IP Outage

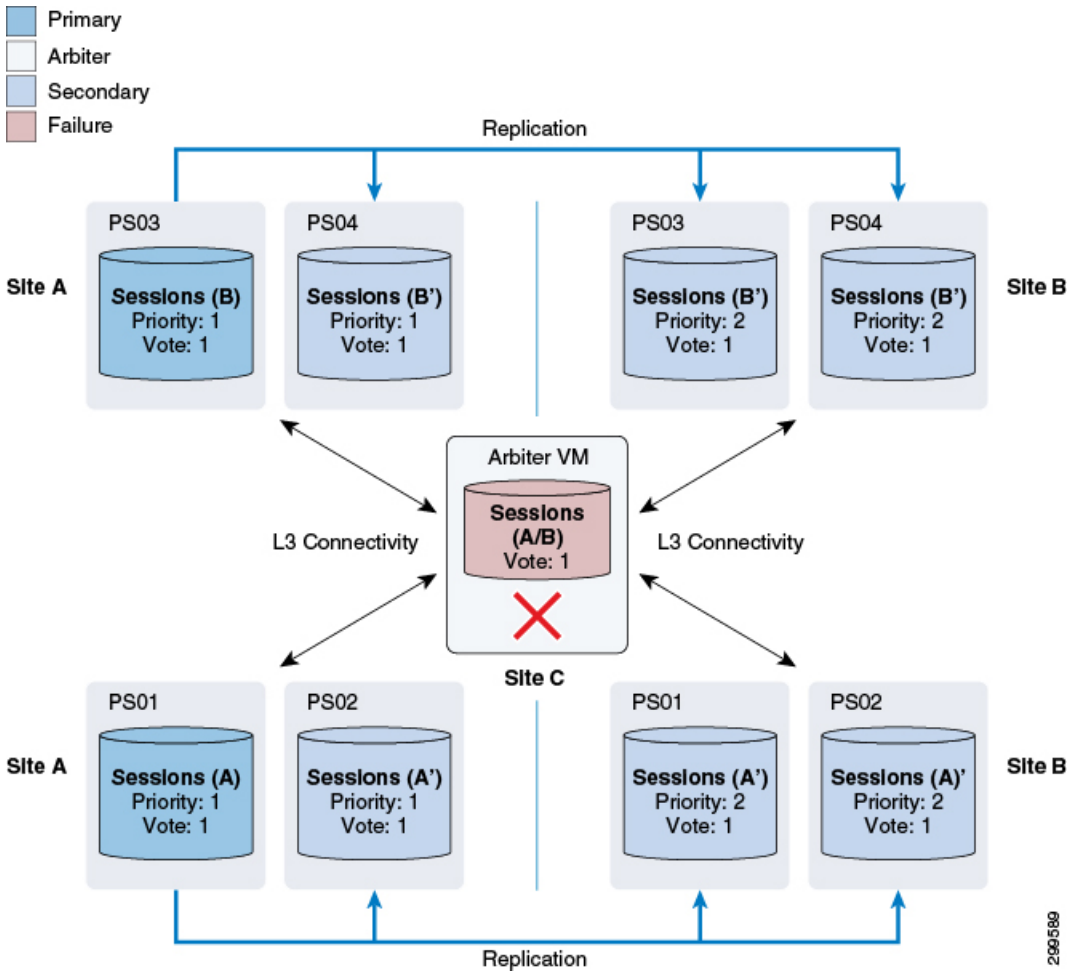
If the network between load balancers and their other communication end points, such as, PGW, fails, CPS will not detect this failure and will continue to operate as it is.

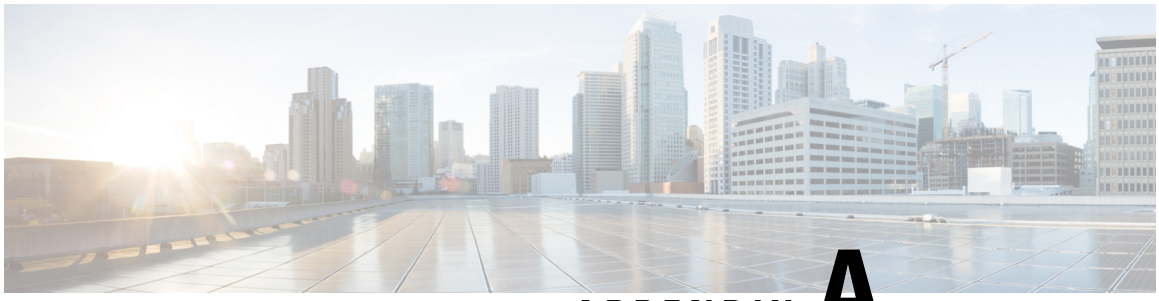
Arbiter Failure

As shown in the figure below, the Arbiter is deployed in a non-redundant manner, as failure of the Arbiter alone does not have any impact on the operation of the Replica Set.

However, a subsequent failure, for example a complete outage of Site A while the Arbiter is down, would result in service interruption as the remaining database instances would not constitute a majority that would allow the election of a new Master database.

Figure 48: Arbiter Failure





APPENDIX A

OpenStack Sample Files - GR

The information in the following sections is for your reference only. You need to modify them according to your requirements.

- Sample Heat Environment File, on page 151
- Sample Heat Template File, on page 153
- Sample YAML Configuration File - site1, on page 178
- Sample YAML Configuration File - site2, on page 186
- Sample Mongo Configuration File - site1, on page 193
- Sample Mongo Configuration File - site2, on page 195
- Sample Mongo GR Configuration File, on page 196
- Sample GR Cluster Configuration File - site1, on page 199
- Sample GR Cluster Configuration File - site2, on page 199
- Sample Set Priority File - site1, on page 199
- Sample Set Priority File - site2, on page 199
- Sample Shard Configuration File - site1, on page 200
- Sample Shard Configuration File - site2, on page 200
- Sample Ring Configuration File, on page 200
- Sample Geo Site Lookup Configuration File - site1, on page 200
- Sample Geo Site Lookup Configuration File - site2, on page 201
- Sample Geo-tagging Configuration File - site1, on page 201
- Sample Geo-tagging Configuration File - site2, on page 201
- Sample Monitor Database Configuration File - site1, on page 201
- Sample Monitor Database Configuration File - site2, on page 202

Sample Heat Environment File

```
# This is an example environment file from os24

parameters:
  cps_iso_image_name: CPS_XXX.iso <----- where, XXX is iso build name.
  base_vm_image_name: base_vm
  cps_az_1: az-1
  cps_az_2: az-2

  internal_net_name: internal
  internal_net_cidr: 192.169.21.0/24
```

OpenStack Sample Files - GR

```
management_net_name: management
management_net_cidr: 192.169.23.0/24
management_net_gateway: 192.169.23.1

gx_net_name: gx
gx_net_cidr: 192.169.22.0/24

external_net_name: external
external_net_cidr: 192.169.24.0/24
external_net_gateway: 192.169.24.1

cluman_flavor_name: cluman
cluman_internal_ip: 192.169.21.10
cluman_management_ip: 192.169.23.10
cluman_external_ip: 192.169.24.10

lb_internal_vip: 192.169.21.21
lb_management_vip: 192.169.23.21
lb_gx_vip: 192.169.22.21
lb_external_vip: 192.169.24.21
lb01_flavor_name: lb01
lb01_internal_ip: 192.169.21.11
lb01_management_ip: 192.169.23.11
lb01_gx_ip: 192.169.22.11
lb01_external_ip: 192.169.24.11
lb02_flavor_name: lb02
lb02_internal_ip: 192.169.21.12
lb02_management_ip: 192.169.23.12
lb02_gx_ip: 192.169.22.12
lb02_external_ip: 192.169.24.12

pcrfclient01_flavor_name: pcrfclient01
pcrfclient01_internal_ip: 192.169.21.19
pcrfclient01_management_ip: 192.169.23.19
pcrfclient01_external_ip: 192.169.24.19
pcrfclient02_flavor_name: pcrfclient02
pcrfclient02_internal_ip: 192.169.21.20
pcrfclient02_management_ip: 192.169.23.20
pcrfclient02_external_ip: 192.169.24.20

qns01_internal_ip: 192.169.21.15
qns01_management_ip: 192.169.23.15
qns01_external_ip: 192.169.24.15

qns02_internal_ip: 192.169.21.16
qns02_management_ip: 192.169.23.16
qns02_external_ip: 192.169.24.16

qns03_internal_ip: 192.169.21.17
qns03_management_ip: 192.169.23.17
qns03_external_ip: 192.169.24.17

qns04_internal_ip: 192.169.21.18
qns04_management_ip: 192.169.23.18
qns04_external_ip: 192.169.24.18

sessionmgr01_internal_ip: 192.169.21.13
sessionmgr01_management_ip: 192.169.23.13
sessionmgr01_external_ip: 192.169.24.13

sessionmgr02_internal_ip: 192.169.21.14
sessionmgr02_management_ip: 192.169.23.14
sessionmgr02_external_ip: 192.169.24.14
```

```
  sessionmgr03_internal_ip: 192.169.21.22
  sessionmgr03_management_ip: 192.169.23.22
  sessionmgr03_external_ip: 192.169.24.22

  sessionmgr04_internal_ip: 192.169.21.23
  sessionmgr04_management_ip: 192.169.23.23
  sessionmgr04_external_ip: 192.169.24.23

  svn01_volume_id: "19d61e3e-a948-46e1-aa38-d953ab98e9a3"
  svn02_volume_id: "3d07bf7f-7a23-43e2-8b93-d705f3bd0619"
  mongo01_volume_id: "23e10db6-0f51-463d-97b9-5b8329f30ec4"
  mongo02_volume_id: "57adb91c-be6e-449e-9f31-8061df726e45"
  mongo03_volume_id: "0e2ebce2-9996-4a6f-96ad-c22f3f873570"
  mongo04_volume_id: "552c311a-1082-4898-bc18-2d959fbefc39"
  cps_iso_volume_id: "023528a2-ac87-4f7c-b868-5ba0346c2673"
```

Sample Heat Template File

Note RADIUS-based policy control is no longer supported in CPS 14.0.0 and later releases as 3GPP Gx Diameter interface has become the industry-standard policy control interface.

```
heat_template_version: 2014-10-16

description: A minimal CPS deployment for big bang deployment

parameters:
#=========================
# Global Paramaters
#=========================
  base_vm_image_name:
    type: string
    label: base vm image name
    description: name of the base vm as imported into glance
  cps_iso_image_name:
    type: string
    label: cps iso image name
    description: name of the cps iso as imported into glance
  cps_install_type:
    type: string
    label: cps installation type (mobile|wifi|mog|pats|arbiter|dra|andsf|escef)
    description: cps installation type (mobile|wifi|mog|arbiter|dra|andsf|escef)
    default: mobile
  cps_az_1:
    type: string
    label: first availability zone
    description: az for "first half" of cluster
    default: nova
  cps_az_2:
    type: string
    label: second availability zone
    description: az for "second half" of cluster
    default: nova

#=========================
# Network Paramaters
#=========================
  internal_net_name:
    type: string
```

OpenStack Sample Files - GR

```
    label: internal network name
    description: name of the internal network
  internal_net_cidr:
    type: string
    label: cps internal cidr
    description: cidr of internal subnet

  management_net_name:
    type: string
    label: management network name
    description: name of the management network
  management_net_cidr:
    type: string
    label: cps management cidr
    description: cidr of management subnet
  management_net_gateway:
    type: string
    label: management network gateway
    description: gateway on management network
    default: ""

  gx_net_name:
    type: string
    label: gx network name
    description: name of the gx network
  gx_net_cidr:
    type: string
    label: cps gx cidr
    description: cidr of gx subnet
  gx_net_gateway:
    type: string
    label: gx network gateway
    description: gateway on gx network
    default: ""

  external_net_name:
    type: string
    label: external network name
    description: name of the external network
  external_net_cidr:
    type: string
    label: cps external cidr
    description: cidr of external subnet
  external_net_gateway:
    type: string
    label: external network gateway
    description: gateway on external network
    default: ""

  cps_secgroup_name:
    type: string
    label: cps secgroup name
    description: name of cps security group
    default: cps_secgroup

#=========================
# Volume Paramaters
#=========================
  mongo01_volume_id:
    type: string
    label: mongo01 volume id
    description: uuid of the mongo01 volume

  mongo02_volume_id:
```

OpenStack Sample Files - GR

```
    type: string
    label: mongo02 volume id
    description: uuid of the mongo02 volume

  mongo03_volume_id:
    type: string
    label: mongo03 volume id
    description: uuid of the mongo03 volume

  mongo04_volume_id:
    type: string
    label: mongo04 volume id
    description: uuid of the mongo04 volume

  svn01_volume_id:
    type: string
    label: svn01 volume id
    description: uuid of the svn01 volume

  svn02_volume_id:
    type: string
    label: svn02 volume id
    description: uuid of the svn02 volume

  cps_iso_volume_id:
    type: string
    label: cps iso volume id
    description: uuid of the cps iso volume

#=========================
# Instance Parameters
#=========================
  cluman_flavor_name:
    type: string
    label: cluman flavor name
    description: flavor cluman vm will use
    default: cluman
  cluman_internal_ip:
    type: string
    label: internal ip of cluster manager
    description: internal ip of cluster manager
  cluman_management_ip:
    type: string
    label: management ip of cluster manager
    description: management ip of cluster manager
  cluman_external_ip:
    type: string
    label: external ip of cluster manager
    description: external ip of cluster manager

  lb_internal_vip:
    type: string
    label: internal vip of load balancer
    description: internal vip of load balancer
  lb_management_vip:
    type: string
    label: management vip of load balancer
    description: management vip of load balancer
  lb_gx_vip:
    type: string
    label: gx ip of load balancer
    description: gx vip of load balancer
  lb_external_vip:
    type: string
```

OpenStack Sample Files - GR

```
  label: external ip of load balancer
  description: external vip of load balancer
lb01_flavor_name:
  type: string
  label: lb01 flavor name
  description: flavor lb01 vms will use
  default: lb01
lb01_internal_ip:
  type: string
  label: internal ip of load balancer
  description: internal ip of load balancer
lb01_management_ip:
  type: string
  label: management ip of load balancer
  description: management ip of load balancer
lb01_gx_ip:
  type: string
  label: gx ip of load balancer
  description: gx ip of load balancer
lb01_external_ip:
  type: string
  label: external ip of load balancer
  description: external ip of load balancer
lb02_flavor_name:
  type: string
  label: lb02 flavor name
  description: flavor lb02 vms will use
  default: lb02
lb02_internal_ip:
  type: string
  label: internal ip of load balancer
  description: internal ip of load balancer
lb02_management_ip:
  type: string
  label: management ip of load balancer
  description: management ip of load balancer
lb02_gx_ip:
  type: string
  label: gx ip of load balancer
  description: gx ip of load balancer
lb02_external_ip:
  type: string
  label: external ip of load balancer lb02
  description: external ip of load balancer lb02

pcrfclient01_flavor_name:
  type: string
  label: pcrfclient01 flavor name
  description: flavor pcrfclient01 vm will use
  default: pcrfclient01
pcrfclient01_internal_ip:
  type: string
  label: internal ip of pcrfclient01
  description: internal ip of pcrfclient01
pcrfclient01_management_ip:
  type: string
  label: management ip of pcrfclient01
  description: management ip of pcrfclient01
pcrfclient01_external_ip:
  type: string
  label: external ip of pcerfclient01
  description: external ip of pcerfclient01

pcrfclient02_flavor_name:
```

```
  type: string
  label: pcrfclient02 flavor name
  description: flavor pcrfclient02 vm will use
  default: pcrfclient02
pcrfclient02_internal_ip:
  type: string
  label: internal ip of pcrfclient02
  description: internal ip of pcrfclient02
pcrfclient02_management_ip:
  type: string
  label: management ip of pcrfclient02
  description: management ip of pcrfclient02
pcrfclient02_external_ip:
  type: string
  label: external ip of pcerfclient02
  description: external ip of pcerfclient02


qns_flavor_name:
  type: string
  label: qns flavor name
  description: flavor qns vms will use
  default: qps
qns01_internal_ip:
  type: string
  label: internal ip of qns01
  description: internal ip of qns01
qns01_management_ip:
  type: string
  label: management ip of qns01
  description: management ip of qns01
qns01_external_ip:
  type: string
  label: external ip of qns01
  description: external ip of qns01

qns02_internal_ip:
  type: string
  label: internal ip of qns02
  description: internal ip of qns02
qns02_management_ip:
  type: string
  label: management ip of qns02
  description: management ip of qns02
qns02_external_ip:
  type: string
  label: external ip of qns02
  description: external ip of qns02

qns03_internal_ip:
  type: string
  label: internal ip of qns03
  description: internal ip of qns03
qns03_management_ip:
  type: string
  label: management ip of qns03
  description: management ip of qns03
qns03_external_ip:
  type: string
  label: external ip of qns03
  description: external ip of qns03

qns04_internal_ip:
  type: string
```

OpenStack Sample Files - GR

```
  label: internal ip of qns04
  description: internal ip of qns04
qns04_management_ip:
  type: string
  label: management ip of qns04
  description: management ip of qns04
qns04_external_ip:
  type: string
  label: external ip of qns04
  description: external ip of qns04

sessionmgr_flavor_name:
  type: string
  label: sessionmgr flavor name
  description: flavor sessionmgr vms will use
  default: sm
sessionmgr01_internal_ip:
  type: string
  label: internal ip of sessionmgr01
  description: internal ip of sessionmgr01
sessionmgr01_management_ip:
  type: string
  label: management ip of sessionmgr01
  description: management ip of sessionmgr01
sessionmgr01_external_ip:
  type: string
  label: external ip of sessionmgr01
  description: external ip of sessionmgr01

sessionmgr02_internal_ip:
  type: string
  label: internal ip of sessionmgr02
  description: internal ip of sessionmgr02
sessionmgr02_management_ip:
  type: string
  label: management ip of sessionmgr02
  description: management ip of sessionmgr02
sessionmgr02_external_ip:
  type: string
  label: external ip of sessionmgr02
  description: external ip of sessionmgr02

sessionmgr03_internal_ip:
  type: string
  label: internal ip of sessionmgr03
  description: external ip of sessionmgr03
sessionmgr03_management_ip:
  type: string
  label: management ip of sessionmgr03
  description: management ip of sessionmgr03
sessionmgr03_external_ip:
  type: string
  label: external ip of sessionmgr03
  description: external ip of sessionmgr03

sessionmgr04_internal_ip:
  type: string
  label: internal ip of sessionmgr04
  description: internal ip of sessionmgr04
sessionmgr04_management_ip:
  type: string
  label: management ip of sessionmgr04
  description: management ip of sessionmgr04
sessionmgr04_external_ip:
```

```
    type: string
    label: external ip of sessionmgr04
    description: external ip of sessionmgr04


resources:
#=========================
# Instances
#=========================

  cluman:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: cluman_flavor_name }
      networks:
        - port: { get_resource: cluman_internal_port }
        - port: { get_resource: cluman_management_port }
        - port: { get_resource: cluman_external_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: cps_iso_volume_id }
      user_data_format: RAW
      user_data: { get_resource: cluman_config }
  cluman_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: cluman_internal_ip }}]
  cluman_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: cluman_management_ip }}]
  cluman_external_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: external_net_name }
      fixed_ips: [{ ip_address: { get_param: cluman_external_ip }}]
  cluman_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
            permissions: "0644"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            permissions: "0644"
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: cluman_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            permissions: "0644"
            content:
              str_replace:
                template: |
```

OpenStack Sample Files - GR

```
              DEVICE=eth1
              BOOTPROTO=none
              NM_CONTROLLED=no
              IPADDR=$ip
              GATEWAY=$gateway
            params:
              $ip: { get_param: cluman_management_ip }
              $gateway: { get_param: management_net_gateway }
      - path: /etc/sysconfig/network-scripts/ifcfg-eth2
        permissions: "0644"
        content:
          str_replace:
            template: |
              DEVICE=eth2
              BOOTPROTO=none
              NM_CONTROLLED=no
              IPADDR=$ip
              GATEWAY=$gateway
            params:
              $ip: { get_param: cluman_external_ip }
              $gateway: { get_param: external_net_gateway }
      - path: /root/.autoinstall.sh
        permissions: "0755"
        content:
          str_replace:
            template: |
              #!/bin/bash
              if [[ -d /mnt/iso ]] && [[ -f /mnt/iso/install.sh ]]; then
              /mnt/iso/install.sh << EOF
              $install_type
              y
              1
              EOF
              fi
            params:
              $install_type: { get_param: cps_install_type }
    mounts:
      - [ /dev/vdb, /mnt/iso, iso9660, "auto,ro", 0, 0 ]
    runcmd:
      - str_replace:
          template: echo $ip installer >> /etc/hosts
          params:
            $ip: { get_param: cluman_internal_ip }
      - str_replace:
          template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
          params:
            $cidr: { get_param: internal_net_cidr }
      - str_replace:
          template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
          params:
            $cidr: { get_param: management_net_cidr }
      - str_replace:
          template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
          params:
            $cidr: { get_param: external_net_cidr }
      - ifdown eth0 && ifup eth0
      - ifdown eth1 && ifup eth1
      - ifdown eth2 && ifup eth2
      - echo HOSTNAME=cluman >> /etc/sysconfig/network

      - hostname cluman
      - /root/.autoinstall.sh

  lb01:
```

```
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: lb01_flavor_name }
    networks:
      - port: { get_resource: lb01_internal_port }
      - port: { get_resource: lb01_management_port }
      - port: { get_resource: lb01_gx_port }
      - port: { get_resource: lb01_external_port }
    user_data_format: RAW
    user_data: { get_resource: lb01_config }
lb01_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_internal_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_internal_vip }
lb01_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_management_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_management_vip }
lb01_gx_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: gx_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_gx_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_gx_vip }
lb01_external_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: external_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_external_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_external_vip }
lb01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=lb01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: lb01_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
```

OpenStack Sample Files - GR

```
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: lb01_management_ip }
                $gateway: { get_param: management_net_gateway }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth2
            content:
              str_replace:
                template: |
                  DEVICE=eth2
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
                  $ip: { get_param: lb01_gx_ip }
                  $gateway: { get_param: gx_net_gateway }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth3
            content:
              str_replace:
                template: |
                  DEVICE=eth3
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
                  $ip: { get_param: lb01_external_ip }
                  $gateway: { get_param: external_net_gateway }
        runcmd:
          - str_replace:
              template: echo $ip installer >> /etc/hosts
              params:
                $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
              params:
                $cidr: { get_param: gx_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth3
              params:
                $cidr: { get_param: external_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - ifdown eth2 && ifup eth2
          - ifdown eth3 && ifup eth3
          - echo HOSTNAME=lb01 >> /etc/sysconfig/network

          - hostname lb01

      lb02:
        type: OS::Nova::Server
        properties:
          availability_zone: { get_param: cps_az_2 }
```

```
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: lb02_flavor_name }
    networks:
      - port: { get_resource: lb02_internal_port }
      - port: { get_resource: lb02_management_port }
      - port: { get_resource: lb02_gx_port }
      - port: { get_resource: lb02_external_port }
    user_data_format: RAW
    user_data: { get_resource: lb02_config }
lb02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: lb02_internal_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_internal_vip }
lb02_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: lb02_management_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_management_vip }
lb02_gx_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: gx_net_name }
    fixed_ips: [{ ip_address: { get_param: lb02_gx_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_gx_vip }
lb02_external_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: external_net_name }
    fixed_ips: [{ ip_address: { get_param: lb02_external_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_external_vip }
lb02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=lb02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: lb02_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
```

OpenStack Sample Files - GR

```
                GATEWAY=$gateway
              params:
                $ip: { get_param: lb02_management_ip }
                $gateway: { get_param: management_net_gateway }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth2
            content:
              str_replace:
                template: |
                  DEVICE=eth2
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
                  $ip: { get_param: lb02_gx_ip }
                  $gateway: { get_param: gx_net_gateway }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth3
            content:
              str_replace:
                template: |
                  DEVICE=eth3
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
                  $ip: { get_param: lb02_external_ip }
                  $gateway: { get_param: external_net_gateway }
        runcmd:
          - str_replace:
              template: echo $ip installer >> /etc/hosts
              params:
                $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
              params:
                $cidr: { get_param: gx_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth3
              params:
                $cidr: { get_param: external_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - ifdown eth2 && ifup eth2
          - ifdown eth3 && ifup eth3
          - echo HOSTNAME=lb02 >> /etc/sysconfig/network

          - hostname lb02

  pcrfclient01:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: pcrfclient01_flavor_name }
```

```
    networks:
      - port: { get_resource: pcrfclient01_internal_port }
      - port: { get_resource: pcrfclient01_management_port }
      - port: { get_resource: pcrfclient01_external_port }
    block_device_mapping:
      - device_name: vdb
        volume_id: { get_param: svn01_volume_id }
    user_data_format: RAW
    user_data: { get_resource: pcrfclient01_config }
pcrfclient01_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient01_internal_ip }}]
pcrfclient01_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient01_management_ip }}]
pcrfclient01_external_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: external_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient01_external_ip }}]
pcrfclient01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=pcrfclient01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: pcrfclient01_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: pcrfclient01_management_ip }
                $gateway: { get_param: management_net_gateway }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth2
          content:
            str_replace:
              template: |
                DEVICE=eth2
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
```

OpenStack Sample Files - GR

```
                $ip: { get_param: pcrfclient01_external_ip }
                $gateway: { get_param: external_net_gateway }
        runcmd:
          - str_replace:
              template: echo $ip installer >> /etc/hosts
              params:
                $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
              params:
                $cidr: { get_param: external_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - ifdown eth2 && ifup eth2
          - echo HOSTNAME=pcrfclient01 >> /etc/sysconfig/network

          - hostname pcrfclient01

  pcrfclient02:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_2 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: pcrfclient02_flavor_name }
      networks:
        - port: { get_resource: pcrfclient02_internal_port }
        - port: { get_resource: pcrfclient02_management_port }
        - port: { get_resource: pcrfclient02_external_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: svn02_volume_id }
      user_data_format: RAW
      user_data: { get_resource: pcrfclient02_config }
  pcrfclient02_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: pcrfclient02_internal_ip }}]
  pcrfclient02_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: pcrfclient02_management_ip }}]
  pcrfclient02_external_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: external_net_name }
      fixed_ips: [{ ip_address: { get_param: pcrfclient02_external_ip }}]
  pcrfclient02_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
          - path: /etc/broadhop.profile
```

```
        content: "NODE_TYPE=pcrfclient02\n"
      - path: /etc/sysconfig/network-scripts/ifcfg-eth0
        content:
          str_replace:
            template: |
              DEVICE=eth0
              BOOTPROTO=none
              NM_CONTROLLED=no
              IPADDR=$ip
            params:
              $ip: { get_param: pcrfclient02_internal_ip }
      - path: /etc/sysconfig/network-scripts/ifcfg-eth1
        content:
          str_replace:
            template: |
              DEVICE=eth1
              BOOTPROTO=none
              NM_CONTROLLED=no
              IPADDR=$ip
              GATEWAY=$gateway
            params:
              $ip: { get_param: pcrfclient02_management_ip }
              $gateway: { get_param: management_net_gateway }
      - path: /etc/sysconfig/network-scripts/ifcfg-eth2
        content:
          str_replace:
            template: |
              DEVICE=eth2
              BOOTPROTO=none
              NM_CONTROLLED=no
              IPADDR=$ip
              GATEWAY=$gateway
            params:
              $ip: { get_param: pcrfclient02_external_ip }
              $gateway: { get_param: external_net_gateway }
    runcmd:
      - str_replace:
          template: echo $ip installer >> /etc/hosts
          params:
            $ip: { get_param: cluman_internal_ip }
      - str_replace:
          template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
          params:
            $cidr: { get_param: internal_net_cidr }
      - str_replace:
          template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
          params:
            $cidr: { get_param: management_net_cidr }
      - str_replace:
          template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
          params:
            $cidr: { get_param: external_net_cidr }
      - ifdown eth0 && ifup eth0
      - ifdown eth1 && ifup eth1
      - ifdown eth2 && ifup eth2
      - echo HOSTNAME=pcrfclient02 >> /etc/sysconfig/network

      - hostname pcrfclient02

qns01:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
```

OpenStack Sample Files - GR

```
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns01_internal_port }
      - port: { get_resource: qns01_external_port }
    user_data_format: RAW
    user_data: { get_resource: qns01_config }
qns01_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns01_internal_ip }}]
qns01_external_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: external_net_name }
    fixed_ips: [{ ip_address: { get_param: qns01_external_ip }}]
qns01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns01_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns01_external_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: external_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - echo HOSTNAME=qns01 >> /etc/sysconfig/network

        - hostname qns01
```

```
qns02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns02_internal_port }
      - port: { get_resource: qns02_external_port }
    user_data_format: RAW
    user_data: { get_resource: qns02_config }
qns02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns02_internal_ip }}]
qns02_external_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: external_net_name }
    fixed_ips: [{ ip_address: { get_param: qns02_external_ip }}]
qns02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns02_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns02_external_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: external_net_cidr }
        - ifdown eth0 && ifup eth0
```

OpenStack Sample Files - GR

```
        - ifdown eth1 && ifup eth1
        - echo HOSTNAME=qns02 >> /etc/sysconfig/network

        - hostname qns02

qns03:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns03_internal_port }
      - port: { get_resource: qns03_external_port }
    user_data_format: RAW
    user_data: { get_resource: qns03_config }
qns03_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns03_internal_ip }}]
qns03_external_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: external_net_name }
    fixed_ips: [{ ip_address: { get_param: qns03_external_ip }}]
qns03_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns03\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns03_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns03_external_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
```

```
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: external_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - echo HOSTNAME=qns03 >> /etc/sysconfig/network

        - hostname qns03

qns04:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns04_internal_port }
      - port: { get_resource: qns04_external_port }
    user_data_format: RAW
    user_data: { get_resource: qns04_config }
qns04_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns04_internal_ip }}]
qns04_external_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: external_net_name }
    fixed_ips: [{ ip_address: { get_param: qns04_external_ip }}]
qns04_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns04\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns04_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns04_external_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
```

OpenStack Sample Files - GR

```
              $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: external_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - echo HOSTNAME=qns04 >> /etc/sysconfig/network

          - hostname qns04

  sessionmgr01:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: sessionmgr_flavor_name }
      networks:
        - port: { get_resource: sessionmgr01_internal_port }
        - port: { get_resource: sessionmgr01_management_port }
        - port: { get_resource: sessionmgr01_external_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: mongo01_volume_id }
      user_data_format: RAW
      user_data: { get_resource: sessionmgr01_config }
  sessionmgr01_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr01_internal_ip }}]
  sessionmgr01_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr01_management_ip }}]
  sessionmgr01_external_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: external_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr01_external_ip }}]
  sessionmgr01_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
          - path: /etc/broadhop.profile
            content: "NODE_TYPE=sessionmgr01\n"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: sessionmgr01_internal_ip }
```

```
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: sessionmgr01_management_ip }
                $gateway: { get_param: management_net_gateway }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth2
          content:
            str_replace:
              template: |
                DEVICE=eth2
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: sessionmgr01_external_ip }
                $gateway: { get_param: external_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
            params:
              $cidr: { get_param: external_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - ifdown eth2 && ifup eth2
        - echo HOSTNAME=sessionmgr01-site2 >> /etc/sysconfig/network

        - hostname sessionmgr01-site2

  sessionmgr02:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_2 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: sessionmgr_flavor_name }
      networks:
        - port: { get_resource: sessionmgr02_internal_port }
        - port: { get_resource: sessionmgr02_management_port }
        - port: { get_resource: sessionmgr02_external_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: mongo02_volume_id }
      user_data_format: RAW
      user_data: { get_resource: sessionmgr02_config }
```

OpenStack Sample Files - GR

```
sessionmgr02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr02_internal_ip }}]
sessionmgr02_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr02_management_ip }}]
sessionmgr02_external_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: external_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr02_external_ip }}]
sessionmgr02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=sessionmgr02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: sessionmgr02_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: sessionmgr02_management_ip }
                $gateway: { get_param: management_net_gateway }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth2
          content:
            str_replace:
              template: |
                DEVICE=eth2
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: sessionmgr02_external_ip }
                $gateway: { get_param: external_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
```

OpenStack Sample Files - GR

```
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
            params:
              $cidr: { get_param: external_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - ifdown eth2 && ifup eth2
        - echo HOSTNAME=sessionmgr02-site2 >> /etc/sysconfig/network

        - hostname sessionmgr02-site2

sessionmgr03:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: sessionmgr_flavor_name }
    networks:
      - port: { get_resource: sessionmgr03_internal_port }
      - port: { get_resource: sessionmgr03_management_port }
      - port: { get_resource: sessionmgr03_external_port }
    block_device_mapping:
      - device_name: vdb
        volume_id: { get_param: mongo03_volume_id }
    user_data_format: RAW
    user_data: { get_resource: sessionmgr03_config }
sessionmgr03_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr03_internal_ip }}]
sessionmgr03_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr03_management_ip }}]
sessionmgr03_external_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: external_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr03_external_ip }}]
sessionmgr03_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=sessionmgr03\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
```

OpenStack Sample Files - GR

```
                params:
                  $ip: { get_param: sessionmgr03_internal_ip }
              - path: /etc/sysconfig/network-scripts/ifcfg-eth1
                content:
                  str_replace:
                    template: |
                      DEVICE=eth1
                      BOOTPROTO=none
                      NM_CONTROLLED=no
                      IPADDR=$ip
                      GATEWAY=$gateway
                    params:
                      $ip: { get_param: sessionmgr03_management_ip }
                      $gateway: { get_param: management_net_gateway }
              - path: /etc/sysconfig/network-scripts/ifcfg-eth2
                content:
                  str_replace:
                    template: |
                      DEVICE=eth2
                      BOOTPROTO=none
                      NM_CONTROLLED=no
                      IPADDR=$ip
                      GATEWAY=$gateway
                    params:
                      $ip: { get_param: sessionmgr03_external_ip }
                      $gateway: { get_param: external_net_gateway }
            runcmd:
              - str_replace:
                  template: echo $ip installer >> /etc/hosts
                  params:
                    $ip: { get_param: cluman_internal_ip }
              - str_replace:
                  template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
                  params:
                    $cidr: { get_param: internal_net_cidr }
              - str_replace:
                  template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
                  params:
                    $cidr: { get_param: management_net_cidr }
              - str_replace:
                  template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
                  params:
                    $cidr: { get_param: external_net_cidr }
              - ifdown eth0 && ifup eth0
              - ifdown eth1 && ifup eth1
              - ifdown eth2 && ifup eth2
              - echo HOSTNAME=sessionmgr03-site2 >> /etc/sysconfig/network

              - hostname sessionmgr03-site2

      sessionmgr04:
        type: OS::Nova::Server
        properties:
          availability_zone: { get_param: cps_az_2 }
          config_drive: "True"
          image: { get_param: base_vm_image_name }
          flavor: { get_param: sessionmgr_flavor_name }
          networks:
            - port: { get_resource: sessionmgr04_internal_port }
            - port: { get_resource: sessionmgr04_management_port }
            - port: { get_resource: sessionmgr04_external_port }
          block_device_mapping:
            - device_name: vdb
              volume_id: { get_param: mongo04_volume_id }
```

```
    user_data_format: RAW
    user_data: { get_resource: sessionmgr04_config }
sessionmgr04_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr04_internal_ip }}]
sessionmgr04_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr04_management_ip }}]
sessionmgr04_external_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: external_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr04_external_ip }}]
sessionmgr04_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=sessionmgr04\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: sessionmgr04_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: sessionmgr04_management_ip }
                $gateway: { get_param: management_net_gateway }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth2
          content:
            str_replace:
              template: |
                DEVICE=eth2
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: sessionmgr04_external_ip }
                $gateway: { get_param: external_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
```

```
- str_replace:
    template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
    params:
      $cidr: { get_param: internal_net_cidr }
- str_replace:
    template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
    params:
      $cidr: { get_param: management_net_cidr }
- str_replace:
    template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
    params:
      $cidr: { get_param: external_net_cidr }
- ifdown eth0 && ifup eth0
- ifdown eth1 && ifup eth1
- ifdown eth2 && ifup eth2
- echo HOSTNAME=sessionmgr04-site2 >> /etc/sysconfig/network

- hostname sessionmgr04-site2
```

Sample YAML Configuration File - site1

Note RADIUS-based policy control is no longer supported in CPS 14.0.0 and later releases as 3GPP Gx Diameter interface has become the industry-standard policy control interface.

```
---
#
#  CPS system configuration
#
#  CPS configuration is a YAML file with all the configuration required
#  to bring up a new installation of CPS.
#
#  This example file lists all possible configuration fields.
#  Fields that are not marked as required can be left out of
#  the configuration. Fields that are not provided will use
#  the default value. If not default is indicated the default
#  is an empty string.

# The version of the configuration file. The installation documentation
# for the version of the CPS you are installing will indicate which
# configuration version you must use.
# REQUIRED
configVersion: 1.0

# Configuration section for CPS hosts
# REQUIRED
hosts:
  # The host section must specify all hosts that are members of the CPS
  # deployment. Host entries consist of the following REQUIRED fields
  #  name: the string to be used as a hostname for the VM
  #  alias: the string to be used in hostname lookup for the VM
  #  interfaces: Network details consisting of the following REQUIRED fields
  #    network: The network name which must match a VLAN name (see below)
  #    ipAddress: The interface address
  # Order of interfaces should be same as your cloud-config.
  # For example, Internal > eth0; Management > eth1; Gx > eth2; External > eth3
  - name: "lb01"
    alias: "lb01"
```

OpenStack Sample Files - GR

```
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.11"
      - network: "Management"
        ipAddress: "192.169.23.11"
      - network: "Gx"
        ipAddress: "192.169.22.11"
      - network: "External"
        ipAddress: "192.169.24.11"
  - name: "lb02"
    alias: "lb02"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.12"
      - network: "Management"
        ipAddress: "192.169.23.12"
      - network: "Gx"
        ipAddress: "192.169.22.12"
      - network: "External"
        ipAddress: "192.169.24.12"
  - name: "sessionmgr01-site1"
    alias: "sessionmgr01"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.13"
      - network: "Management"
        ipAddress: "192.169.23.13"
      - network: "External"
        ipAddress: "192.169.24.13"
  - name: "sessionmgr02-site1"
    alias: "sessionmgr02"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.14"
      - network: "Management"
        ipAddress: "192.169.23.14"
      - network: "External"
        ipAddress: "192.169.24.14"
  - name: "sessionmgr03-site1"
    alias: "sessionmgr03"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.22"
      - network: "Management"
        ipAddress: "192.169.23.22"
      - network: "External"
        ipAddress: "192.169.24.22"
  - name: "sessionmgr04-site1"
    alias: "sessionmgr04"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.23"
      - network: "Management"
        ipAddress: "192.169.23.23"
      - network: "External"
        ipAddress: "192.169.24.23"
  - name: "qns01"
    alias: "qns01"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.15"
      - network: "External"
        ipAddress: "192.169.24.15"
  - name: "qns02"
```

OpenStack Sample Files - GR

```
    alias: "qns02"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.16"
      - network: "External"
        ipAddress: "192.169.24.16"
  - name: "qns03"
    alias: "qns03"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.17"
      - network: "External"
        ipAddress: "192.169.24.17"
  - name: "qns04"
    alias: "qns04"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.18"
      - network: "External"
        ipAddress: "192.169.24.18"
  - name: "pcrfclient01"
    alias: "pcrfclient01"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.19"
      - network: "Management"
        ipAddress: "192.169.23.19"
      - network: "External"
        ipAddress: "192.169.24.19"
  - name: "pcrfclient02"
    alias: "pcrfclient02"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.20"
      - network: "Management"
        ipAddress: "192.169.23.20"
      - network: "External"
        ipAddress: "192.169.24.20"

# Configuration section for CPS VLANs
# REQUIRED
vlans:
  # VLAN entries consist of the following REQUIRED fields
  #  name: The VLAN name. This name must be used in the "network" field
  #        host interfaces (see above)
  #  vipAlias: Hostname associated with the vip
  #  vip: Virtual IP used no this network, if any.
  #  guestNic: The name of the interface specified in the host cloud config
  #            or the Heat definition.
  #
  - name: "Internal"
    vipAlias: "lbvip02"
    vip: "192.169.21.21"
  - name: "Management"
    vipAlias: "lbvip01"
    vip: "192.169.23.21"
  - name: "Gx"
    vipAlias: "gxvip"
    vip: "192.169.22.21"
  - name: "External"
    vipAlias: "exvip"
    vip: "192.169.24.21"

# Configuration section for hosts not configured in the hosts section above.
```

```
# REQUIRED
additionalHosts:
  # additionalHosts entries consist of the following REQUIRED fields
  #  name: The hostname
  #  alias: The string to be used in the etc/host file.
  #  ipAddress: The IP address to use in the etc/host file.
  #
  - name: "lbvip01"
    ipAddress: "192.169.23.21"
    alias: "lbvip01"
  - name: "lbvip02"
    ipAddress: "192.169.21.21"
    alias: "lbvip02"
  - name: "diam-int1-vip"
    ipAddress: "192.169.22.21"
    alias: "gxvip"
  - name: "arbitervip"
    ipAddress: "192.169.21.40"
    alias: "arbitervip"
  - name: "cluman-site2"
    alias: "cluman-site2"
    ipAddress: "192.169.24.50"
  - name: "sessionmgr01-site2"
    alias: "psessionmgr01"
    ipAddress: "192.169.24.60"
  - name: "sessionmgr02-site2"
    alias: "psessionmgr02"
    ipAddress: "192.169.24.61"
  - name: "sessionmgr03-site2"
    alias: "psessionmgr03"
    ipAddress: "192.169.24.66"
  - name: "sessionmgr04-site2"
    alias: "psessionmgr04"
    ipAddress: "192.169.24.67"
  - name: "arbiter"
    alias: "arbiter-site3"
    ipAddress: "192.169.24.90"


# Configuration section for general configuration items.
# REQUIRED
config:
  # Do not change. See install documentation for details.
  # default: sys_user_0
  qpsUser: "sys_user_0"

  # Do not change. See install documentation for details.
  # default: disabled
  selinuxState: "disabled"

  # Do not change. See install documentation for details.
  # default: targeted
  selinuxType: "targeted"

  # See install documentation for details.
  # default: broadhop
  broadhopVar: "broadhop"

  # Set true to enable TACACS+ authentication.
  # default: FALSE
  tacacsEnabled: "FALSE"

  # The IP Address of the TACACS+ server
  tacacsServer: "127.0.0.1"
```

OpenStack Sample Files - GR

```
# The password/secret of the TACACS+ server.
tacacsSecret: "CPE1704TKS"

# A set of SNMP Network Management Stations.
# NMS can be specified as IP addresses or IP
# addresses. Entries are space separated.
# Hostnames must also be specified in Additional
# Host configuration.
# See install documentation for details.
nmsManagers:

# Low Memory alert threshold %.
# default: 0.1 (10% free)
freeMemPer: "0.1"

# A space separated set of protocol:hostname:port
# entries. UDP is the only supported protocol.
# Example:
# upd:corporate_syslog_ip:514 udp:corporate_syslog_ip2:514
syslogManagers:

# A comma separated set of port values.
# This must match values in the syslog_managers_list.
# default: 514
syslogManagersPorts: "514"

# Port value for the rsyslog proxy server to listen
# for incoming connections
# default: 6515
logbackSyslogDaemonPort: "6515"

# IP address value used in the
# /etc/broadhop/controlcenter/logback.xml
# on the pcrfclient.
# default: lbvip02
logbackSyslogDaemonAddr: "lbvip02"


# High CPU alert threshold.
# The system will alert whenever the usage is
# higher than this value.
# default: 80
cpuUsageAlertThreshold: "80"

# Clear High CPU Trap threshold.
# The system will generate a clear trap when a
# High CPU trap has been generated and the CPU
# usage is lower than this value.
# default: 40
cpuUsageClearThreshold: "40"

# The number of 5 sec intervals to wait between
# checking the CPU usage.
# default: 12 (60 seconds)
cpuUsageTrapIntervalCycle: "12"

# The SNMP trap community string.
snmpTrapCommunity: "broadhop"

#The SNMP read community string.
snmpRoCommunity: "broadhop"

#
```

OpenStack Sample Files - GR

```
  monQnsLb:

  # Enables or disables linux firewall on all VMs (IPtables).
  # default: disabled
  firewallState: "disabled"

  # Users
  # There are different categories of users specified for the CPS.
  # All users have the following fields:
  #
  #  name: The user name. REQUIRED
  #  password: The password for the user. REQUIRED
  #            The password will need to be either in cleartext or
  #            encrypted. Please refer to Install documentation for details.
  #  groups: The groups for the user. Groups are specified as a list
  #          of group names.

  # System Users
  # Note that there must be a system use named sys_user_0
  sysUsers:
    - name: "qns"
      password:
"$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTkCIC3QW5hkd6P/Sl3OD8qFHn1aYHxce1"

      groups:
        - pwauth

    - name: "qns-svn"
      password:
"$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTkCIC3QW5hkd6P/Sl3OD8qFHn1aYHxce1"

    - name: "qns-ro"
      password:
"$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTkCIC3QW5hkd6P/Sl3OD8qFHn1aYHxce1"


  # Hypervisor Users
  hvUsers:
    - name: "root"
      password: "cisco123"

  # Other Users for the CPS
  # e.g. Control Center Users
  additionalUsers:
    - name: "admin"
      password: "qns123"
      groups:
        - qns


# Configuration section for feature licenses
# REQUIRED
licenses:
  # Licenses have the following required fields:
  # feature: The name of the feature license.
  # license: The license key for the feature.
  # - feature: "feature 1  Name"
  #   license: "license 1 key string"
   - feature: "MOBILE_CORE"
     license:
"25D220C6817CD63603D72ED51C811F9B7CB093A53B5CE6FB04FF6C5C6A21ED1962F0491D4EED4441D826F1BC110B05EE35B78CF43B8B8B7A8127B4545538E365"

   - feature: "RADIUS_AUTH"
     license:
```

OpenStack Sample Files - GR

```
"118D767CE11EC2CB1E3AAA846A916FA57CB093A53B5CE6FB04FF6C5C6A21ED1962F0491D4EED4441D826F1BC110B05EE35B78CF43B8B8B7A8127B4545538E365"

# Configuration section for mongo replica sets.
# REQUIRED
replicaSets:
  #
  # Mongo replica sets have the following REQUIRED fields
  # <Mongo Set Identifier> : The database for which the replica
  #                          set is being created.
  #   setName: The name of the replica set
  #   oplogSize: Mongo Oplog size
  #   arbiter: The Arbiter hosthame and port
  #   arbiterDataPath: The data directory on the arbiter VM
  #   primaryMembers: List of primaryMembers for the replica set. Each list element
  #            will be a session manager hostname:port
  #   dataPath: The data directory path on the session manager VMs
  - title: SESSION-SET1
    setName: set01
    oplogSize: 1024
    arbiter: arbiter-site3:27717
    arbiterDataPath: /var/data/sessions.1
    siteId: "SITE1"
    members:
      - sessionmgr02-site1:27717
      - sessionmgr01-site1:27717
    dataPath: /var/data/sessions.1/set01
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
    shardCount: "4"
    hotStandBy: "false"
    seeds: "sessionmgr01:sessionmgr02:27717"
  - title: SESSION-SET2
    setName: set07
    oplogSize: 1024
    arbiter: arbiter-site3:27722
    arbiterDataPath: /var/data/sessions.7
    siteId: "SITE1"
    members:
      - sessionmgr03-site1:27722
      - sessionmgr04-site1:27722
    dataPath: /var/data/sessions.7
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
    shardCount: "4"
    hotStandBy: "true"
    seeds: "sessionmgr03:sessionmgr04:27722"
  - title: BALANCE-SET1
    setName: set02
    oplogSize: 1024
    arbiter: arbiter-site3:27718
    arbiterDataPath: /var/data/sessions.2
    siteId: "SITE1"
    members:
      - sessionmgr01-site1:27718
      - sessionmgr02-site1:27718
    dataPath: /var/data/sessions.2
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
  - title: REPORTING-SET1
    setName: set03
    oplogSize: 1024
    arbiter: arbiter-site3:27719
    arbiterDataPath: /var/data/sessions.3
    siteId: "SITE1"
```

OpenStack Sample Files - GR

```
    members:
      - sessionmgr03-site1:27719
      - sessionmgr04-site1:27719
    dataPath: /var/data/sessions.3
  - title: SPR-SET1
    setName: set04
    oplogSize: 1024
    arbiter: arbiter-site3:27720
    arbiterDataPath: /var/data/sessions.4
    siteId: "SITE1"
    members:
      - sessionmgr01-site1:27720
      - sessionmgr02-site1:27720
    dataPath: /var/data/sessions.4
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
  - title: AUDIT-SET1
    setName: set05
    oplogSize: 1024
    arbiter: arbiter-site3:27017
    arbiterDataPath: /var/data/sessions.5
    siteId: "SITE1"
    members:
      - sessionmgr03-site1:27017
      - sessionmgr04-site1:27017
    dataPath: /var/data/sessions.5
  - title: ADMIN-SET1
    setName: set06
    oplogSize: 1024
    arbiter: arbiter-site3:27721
    arbiterDataPath: /var/data/sessions.6
    siteId: "SITE1"
    members:
      - sessionmgr01-site1:27721
      - sessionmgr02-site1:27721
    dataPath: /var/data/sessions.6
applicationConfig:
    policyServerConfig:
       geoSiteName: "SITE1"
       clusterId: "Cluster-SITE1"
       siteId: "SITE1"
       remoteSiteId: "SITE2"
       heartBeatMonitorThreadSleepMS: "500"
       mongodbupdaterConnectTimeoutMS: "1000"
       mongodbupdaterSocketTimeoutMS: "1000"
       dbConnectTimeout: "1200"
       threadMaxWaitTime: "1200"
       dbSocketTimeout: "600"
       remoteLockingOff: ""
       apirouterContextPath: ""
       uaContextPath: ""
       balanceDbs: ""
       clusterPeers: ""
       isGeoHaEnabled: "true"
       geoHaSessionLookupType: "realm"
       enableReloadDict: "true"
       sprLocalGeoSiteTag: "SITE1"
       balanceLocalGeoSiteTag: "SITE1"
       sessionLocalGeoSiteTag: "SITE1"
       deploymentType: "GR"
```

Sample YAML Configuration File - site2

Note RADIUS-based policy control is no longer supported in CPS 14.0.0 and later releases as 3GPP Gx Diameter interface has become the industry-standard policy control interface.

```
---
#
#  CPS system configuration
#
#  CPS configuration is a YAML file with all the configuration required
#  to bring up a new installation of CPS.
#
#  This example file lists all possible configuration fields.
#  Fields that are not marked as required can be left out of
#  the configuration. Fields that are not provided will use
#  the default value. If not default is indicated the default
#  is an empty string.

# The version of the configuration file. The installation documentation
# for the version of the CPS you are installing will indicate which
# configuration version you must use.
# REQUIRED
configVersion: 1.0

# Configuration section for CPS hosts
# REQUIRED
hosts:
  # The host section must specify all hosts that are members of the CPS
  # deployment. Host entries consist of the following REQUIRED fields
  #  name: the string to be used as a hostname for the VM
  #  alias: the string to be used in hostname lookup for the VM
  #  interfaces: Network details consisting of the following REQUIRED fields
  #    network: The network name which must match a VLAN name (see below)
  #    ipAddress: The interface address
  # Order of interfaces should be same as your cloud-config.
  # For example, Internal > eth0; Management > eth1; Gx > eth2; External > eth3
  - name: "lb01"
    alias: "lb01"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.52"
      - network: "Management"
        ipAddress: "192.169.23.52"
      - network: "Gx"
        ipAddress: "192.169.22.52"
      - network: "External"
        ipAddress: "192.169.24.52"
  - name: "lb02"
    alias: "lb02"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.53"
      - network: "Management"
        ipAddress: "192.169.23.53"
      - network: "Gx"
        ipAddress: "192.169.22.53"
      - network: "External"
        ipAddress: "192.169.24.53"
```

```
- name: "sessionmgr01-site2"
  alias: "sessionmgr01"
  interfaces:
    - network: "Internal"
      ipAddress: "192.169.21.60"
    - network: "Management"
      ipAddress: "192.169.23.60"
    - network: "External"
      ipAddress: "192.169.24.60"
- name: "sessionmgr02-site2"
  alias: "sessionmgr02"
  interfaces:
    - network: "Internal"
      ipAddress: "192.169.21.61"
    - network: "Management"
      ipAddress: "192.169.23.61"
    - network: "External"
      ipAddress: "192.169.24.61"
- name: "sessionmgr03-site2"
  alias: "sessionmgr03"
  interfaces:
    - network: "Internal"
      ipAddress: "192.169.21.66"
    - network: "Management"
      ipAddress: "192.169.23.66"
    - network: "External"
      ipAddress: "192.169.24.66"
- name: "sessionmgr04-site2"
  alias: "sessionmgr04"
  interfaces:
    - network: "Internal"
      ipAddress: "192.169.21.67"
    - network: "Management"
      ipAddress: "192.169.23.67"
    - network: "External"
      ipAddress: "192.169.24.67"
- name: "qns01"
  alias: "qns01"
  interfaces:
    - network: "Internal"
      ipAddress: "192.169.21.56"
    - network: "External"
      ipAddress: "192.169.24.56"
- name: "qns02"
  alias: "qns02"
  interfaces:
    - network: "Internal"
      ipAddress: "192.169.21.57"
    - network: "External"
      ipAddress: "192.169.24.57"
- name: "qns03"
  alias: "qns03"
  interfaces:
    - network: "Internal"
      ipAddress: "192.169.21.58"
    - network: "External"
      ipAddress: "192.169.24.58"
- name: "qns04"
  alias: "qns04"
  interfaces:
    - network: "Internal"
      ipAddress: "192.169.21.59"
    - network: "External"
      ipAddress: "192.169.24.59"
```

OpenStack Sample Files - GR

```
  - name: "pcrfclient01"
    alias: "pcrfclient01"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.54"
      - network: "Management"
        ipAddress: "192.169.23.54"
      - network: "External"
        ipAddress: "192.169.24.54"
  - name: "pcrfclient02"
    alias: "pcrfclient02"
    interfaces:
      - network: "Internal"
        ipAddress: "192.169.21.55"
      - network: "Management"
        ipAddress: "192.169.23.55"
      - network: "External"
        ipAddress: "192.169.24.55"

# Configuration section for CPS VLANs
# REQUIRED
vlans:
  # VLAN entries consist of the following REQUIRED fields
  #  name: The VLAN name. This name must be used in the "network" field
  #        host interfaces (see above)
  #  vipAlias: Hostname associated with the vip
  #  vip: Virtual IP used no this network, if any.
  #  guestNic: The name of the interface specified in the host cloud config
  #            or the Heat definition.
  #
  - name: "Internal"
    vipAlias: "lbvip02"
    vip: "192.169.21.51"
  - name: "Management"
    vipAlias: "lbvip01"
    vip: "192.169.23.51"
  - name: "Gx"
    vipAlias: "gxvip"
    vip: "192.169.22.51"
  - name: "External"
    vipAlias: "exvip"
    vip: "192.169.24.51"

# Configuration section for hosts not configured in the hosts section above.
# REQUIRED
additionalHosts:
  # additionalHosts entries consist of the following REQUIRED fields
  #  name: The hostname
  #  alias: The string to be used in the etc/host file.
  #  ipAddress: The IP address to use in the etc/host file.
  #
  - name: "lbvip01"
    ipAddress: "192.169.23.51"
    alias: "lbvip01"
  - name: "lbvip02"
    ipAddress: "192.169.21.51"
    alias: "lbvip02"
  - name: "diam-int1-vip"
    ipAddress: "192.169.22.51"
    alias: "gxvip"
  - name: "arbitervip"
    ipAddress: "192.169.21.70"
    alias: "arbitervip"
  - name: "cluman-site2"
```

```
    alias: "cluman-site2"
    ipAddress: "192.169.24.50"
  - name: "sessionmgr01-site1"
    alias: "pessionmgr01"
    ipAddress: "192.169.24.13"
  - name: "sessionmgr02-site1"
    alias: "pessionmgr02"
    ipAddress: "192.169.24.14"
  - name: "sessionmgr03-site1"
    alias: "pessionmgr03"
    ipAddress: "192.169.24.22"
  - name: "sessionmgr04-site1"
    alias: "pessionmgr04"
    ipAddress: "192.169.24.23"
  - name: "arbiter"
    alias: "arbiter-site3"
    ipAddress: "192.169.24.90"
# Configuration section for general configuration items.
# REQUIRED
config:
  # Do not change. See install documentation for details.
  # default: sys_user_0
  qpsUser: "sys_user_0"

  # Do not change. See install documentation for details.
  # default: disabled
  selinuxState: "disabled"

  # Do not change. See install documentation for details.
  # default: targeted
  selinuxType: "targeted"

  # See install documentation for details.
  # default: broadhop
  broadhopVar: "broadhop"

  # Set true to enable TACACS+ authentication.
  # default: FALSE
  tacacsEnabled: "FALSE"

  # The IP Address of the TACACS+ server
  tacacsServer: "127.0.0.1"

  # The password/secret of the TACACS+ server.
  tacacsSecret: "CPE1704TKS"

  # A set of SNMP Network Management Stations.
  # NMS can be specified as IP addresses or IP
  # addresses. Entries are space separated.
  # Hostnames must also be specified in Additional
  # Host configuration.
  # See install documentation for details.
  nmsManagers:

  # Low Memory alert threshold %.
  # default: 0.1 (10% free)
  freeMemPer: "0.1"

  # A space separated set of protocol:hostname:port
  # entries. UDP is the only supported protocol.
  # Example:
  # upd:corporate_syslog_ip:514 udp:corporate_syslog_ip2:514
  syslogManagers:
```

OpenStack Sample Files - GR

```
# A comma separated set of port values.
# This must match values in the syslog_managers_list.
# default: 514
syslogManagersPorts: "514"

# Port value for the rsyslog proxy server to listen
# for incoming connections
# default: 6515
logbackSyslogDaemonPort: "6515"

# IP address value used in the
# /etc/broadhop/controlcenter/logback.xml
# on the pcrfclient.
# default: lbvip02
logbackSyslogDaemonAddr: "lbvip02"


# High CPU alert threshold.
# The system will alert whenever the usage is
# higher than this value.
# default: 80
cpuUsageAlertThreshold: "80"

# Clear High CPU Trap threshold.
# The system will generate a clear trap when a
# High CPU trap has been generated and the CPU
# usage is lower than this value.
# default: 40
cpuUsageClearThreshold: "40"

# The number of 5 sec intervals to wait between
# checking the CPU usage.
# default: 12 (60 seconds)
cpuUsageTrapIntervalCycle: "12"

# The SNMP trap community string.
snmpTrapCommunity: "broadhop"

#The SNMP read community string.
snmpRoCommunity: "broadhop"

#
monQnsLb:

# Enables or disables linux firewall on all VMs (IPtables).
# default: disabled
firewallState: "disabled"

# Users
# There are different categories of users specified for the CPS.
# All users have the following fields:
#
#  name: The user name. REQUIRED
#  password: The password for the user. REQUIRED
#            The password will need to be either in cleartext or
#            encrypted. Please refer to Install documentation for details.
#  groups: The groups for the user. Groups are specified as a list
#          of group names.

# System Users
# Note that there must be a system use named sys_user_0
sysUsers:
  - name: "qns"
    password:
```

OpenStack Sample Files - GR

```
"$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTkCIC3QW5hkd6P/Sl3OD8qFHn1aYHxce1"

      groups:
        - pwauth

    - name: "qns-svn"
      password:
"$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTkCIC3QW5hkd6P/Sl3OD8qFHn1aYHxce1"

    - name: "qns-ro"
      password:
"$6$HtEnOu7S$8kkHDFJtAZtJXnhRPrPFI8KAlHFch41OJ405OnCCqO0CFuRmexvCRTkCIC3QW5hkd6P/Sl3OD8qFHn1aYHxce1"


  # Hypervisor Users
  hvUsers:
    - name: "root"
      password: "cisco123"

  # Other Users for the CPS
  # e.g. Control Center Users
  additionalUsers:
    - name: "admin"
      password: "qns123"
      groups:
        - qns


# Configuration section for feature licenses
# REQUIRED
licenses:
  # Licenses have the following required fields:
  # feature: The name of the feature license.
  # license: The license key for the feature.
  # - feature: "feature 1  Name"
  #   license: "license 1 key string"
   - feature: "MOBILE_CORE"
     license:
"25D220C6817CD63603D72ED51C811F9B7CB093A53B5CE6FB04FF6C5C6A21ED1962F0491D4EED4441D826F1BC110B05EE35B78CF43B8B8B7A8127B4545538E365"

   - feature: "RADIUS_AUTH"
     license:
"118D767CE11EC2CB1E3AAA846A916FA57CB093A53B5CE6FB04FF6C5C6A21ED1962F0491D4EED4441D826F1BC110B05EE35B78CF43B8B8B7A8127B4545538E365"

# Configuration section for mongo replica sets.
# REQUIRED
replicaSets:
  #
  # Mongo replica sets have the following REQUIRED fields
  # <Mongo Set Identifier> : The database for which the replica
  #                          set is being created.
  #   setName: The name of the replica set
  #   oplogSize: Mongo Oplog size
  #   arbiter: The Arbiter hosthame and port
  #   arbiterDataPath: The data directory on the arbiter VM
  #   members: List of members for the replica set. Each list element
  #            will be a session manager hostname:port
  #   dataPath: The data directory path on the session manager VMs
  - title: SESSION-SET63
    setName: set63
    oplogSize: 1024
    arbiter: arbiter-site3:27763
    arbiterDataPath: /var/data/sessions.1/set63
    siteId: "SITE2"
```

OpenStack Sample Files - GR

```
    members:
      - sessionmgr01-site2:27763
      - sessionmgr02-site2:27763
    dataPath: /var/data/sessions.63
    primaryMembersTag: "SITE2"
    secondaryMembersTag: "SITE1"
    shardCount: "4"
    seeds: "sessionmgr01:sessionmgr02:27763"
  - title: SESSION-SET68
    setName: set68
    oplogSize: 1024
    arbiter: arbiter-site3:27768
    arbiterDataPath: /var/data/sessions.68
    siteId: "SITE2"
    members:
      - sessionmgr03-site2:27768
      - sessionmgr04-site2:27768
    dataPath: /var/data/sessions.68
    primaryMembersTag: "SITE2"
    secondaryMembersTag: "SITE1"
    shardCount: "4"
    hotStandBy: "true"
    seeds: "sessionmgr03:sessionmgr04:27768"
  - title: BALANCE-SET64
    setName: set64
    oplogSize: 1024
    arbiter: arbiter-site3:27764
    arbiterDataPath: /var/data/sessions.64
    siteId: "SITE2"
    members:
      - sessionmgr01-site2:27764
      - sessionmgr02-site2:27764
    dataPath: /var/data/sessions.64
    primaryMembersTag: "SITE2"
    secondaryMembersTag: "SITE1"
  - title: REPORTING-SET66
    setName: set66
    oplogSize: 1024
    arbiter: arbiter-site3:27766
    arbiterDataPath: /var/data/sessions.66
    siteId: "SITE2"
    members:
      - sessionmgr03-site2:27719
      - sessionmgr04-site2:27719
    dataPath: /var/data/sessions.66
  - title: SPR-SET67
    setName: set67
    oplogSize: 1024
    arbiter: arbiter-site3:27767
    arbiterDataPath: /var/data/sessions.67
    siteId: "SITE2"
    members:
      - sessionmgr01-site2:27767
      - sessionmgr02-site2:27767
    dataPath: /var/data/sessions.67
    primaryMembersTag: "SITE2"
    secondaryMembersTag: "SITE1"
  - title: AUDIT-SET65
    setName: set65
    oplogSize: 1024
    arbiter: arbiter-site3:27765
    arbiterDataPath: /var/data/sessions.65
    siteId: "SITE2"
    members:
```

```
      - sessionmgr03-site2:27017
      - sessionmgr04-site2:27017
    dataPath: /var/data/sessions.65
  - title: ADMIN-SET2
    setName: set69
    oplogSize: 1024
    arbiter: arbiter-site3:27769
    arbiterDataPath: /var/data/sessions.69
    siteId: "SITE2"
    members:
    - sessionmgr01-site2:27769
    - sessionmgr02-site2:27769
    dataPath: /var/data/sessions.69

applicationConfig:
  policyServerConfig:
    geoSiteName: "SITE2"
    clusterId: "Cluster-SITE2"
    siteId: "SITE2"
    remoteSiteId: "SITE1"
    heartBeatMonitorThreadSleepMS: "500"
    mongodbupdaterConnectTimeoutMS: "1000"
    mongodbupdaterSocketTimeoutMS: "1000"
    dbConnectTimeout: "1200"
    threadMaxWaitTime: "1200"
    dbSocketTimeout: "600"
    remoteLockingOff: ""
    apirouterContextPath: ""
    uaContextPath: ""
    balanceDbs: ""
    clusterPeers: ""
    isGeoHaEnabled: "true"
    geoHaSessionLookupType: "realm"
    enableReloadDict: "true"
    sprLocalGeoSiteTag: "SITE2"
    balanceLocalGeoSiteTag: "SITE2"
    sessionLocalGeoSiteTag: "SITE2"
    deploymentType: "GR"
```

Sample Mongo Configuration File - site1

```
---
- title: "SESSION-SET1"
  setName: "set01"
  oplogSize: "1024"
  arbiter: "arbiter-site3:27717"
  arbiterDataPath: "/var/data/sessions.1"
  primaryMembers:
  - "sessionmgr02-site1:27717"
  - "sessionmgr01-site1:27717"
  secondaryMembers:
  - "sessionmgr02-site2:27717"
  - "sessionmgr01-site2:27717"
  dataPath: "/var/data/sessions.1/set01"
  hotStandBy: "false"
  shardCount: "4"
  seeds: "sessionmgr01:sessionmgr02:27717"
  primaryMembersTag: "SITE1"
  secondaryMembersTag: "SITE2"
  siteId: "SITE1"
- title: "SESSION-SET2"
  setName: "set07"
```

OpenStack Sample Files - GR

```
    oplogSize: "1024"
    arbiter: "arbiter-site3:27722"
    arbiterDataPath: "/var/data/sessions.7"
    members:
    - "sessionmgr03-site1:27722"
    - "sessionmgr04-site1:27722"
    dataPath: "/var/data/sessions.7"
    hotStandBy: "true"
    shardCount: "4"
    seeds: "sessionmgr03:sessionmgr04:27722"
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
    siteId: "SITE1"
  - title: "BALANCE-SET1"
    setName: "set02"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27718"
    arbiterDataPath: "/var/data/sessions.2"
    primaryMembers:
    - "sessionmgr01-site1:27718"
    - "sessionmgr02-site1:27718"
    secondaryMembers:
    - "sessionmgr01-site2:27718"
    - "sessionmgr02-site2:27718"
    dataPath: "/var/data/sessions.2"
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
    siteId: "SITE1"
  - title: "REPORTING-SET1"
    setName: "set03"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27719"
    arbiterDataPath: "/var/data/sessions.3"
    members:
    - "sessionmgr03-site1:27719"
    - "sessionmgr04-site1:27719"
    dataPath: "/var/data/sessions.3"
    siteId: "SITE1"
  - title: "SPR-SET1"
    setName: "set04"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27720"
    arbiterDataPath: "/var/data/sessions.4"
    primaryMembers:
    - "sessionmgr01-site1:27720"
    - "sessionmgr02-site1:27720"
    secondaryMembers:
    - "sessionmgr01-site2:27720"
    - "sessionmgr02-site2:27720"
    dataPath: "/var/data/sessions.4"
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
    siteId: "SITE1"
  - title: "AUDIT-SET1"
    setName: "set05"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27017"
    arbiterDataPath: "/var/data/sessions.5"
    members:
    - "sessionmgr03-site1:27017"
    - "sessionmgr04-site1:27017"
    dataPath: "/var/data/sessions.5"
    siteId: "SITE1"
  - title: "ADMIN-SET1"
```

```
  setName: "set06"
  oplogSize: "1024"
  arbiter: "arbiter-site3:27721"
  arbiterDataPath: "/var/data/sessions.6"
  primaryMembers:
  - "sessionmgr01-site1:27721"
  - "sessionmgr02-site1:27721"
  secondaryMembers:
  - "sessionmgr01-site2:27721"
  - "sessionmgr02-site2:27721"
  dataPath: "/var/data/sessions.6"
  siteId: "SITE1"
```

Sample Mongo Configuration File - site2

```
- title: "SESSION-SET63"
  setName: "set63"
  oplogSize: "1024"
  arbiter: "arbiter:27763"
  arbiterDataPath: "/var/data/sessions.63"
  primaryMembers:
  - "sessionmgr01-site2:27763"
  - "sessionmgr02-site2:27763"
  secondaryMembers:
  - "sessionmgr01-site1:27763"
  - "sessionmgr02-site1:27763"
  dataPath: "/var/data/sessions.1/set63"
  secondaryMembersTag: "SITE1"
  primaryMembersTag: "SITE2"
  siteId: "SITE2"
  shardCount: "4"
  seeds: "sessionmgr01:sessionmgr02:27763"
- title: "SESSION-SET68"
  setName: "set68"
  oplogSize: "1024"
  arbiter: "arbiter:27768"
  arbiterDataPath: "/var/data/sessions.68"
  primaryMembers:
  - "sessionmgr03-site2:27768"
  - "sessionmgr04-site2:27768"
  secondaryMembers:
  - "sessionmgr03-site1:27768"
  - "sessionmgr04-site1:27768"
  dataPath: "/var/data/sessions.68"
  primaryMembersTag: "SITE2"
  secondaryMembersTag: "SITE1"
  hotStandBy: "true"
  shardCount: "4"
  seeds: "sessionmgr03:sessionmgr04:27768"
  siteId: "SITE2"
- title: "BALANCE-SET64"
  setName: "set64"
  oplogSize: "1024"
  arbiter: "arbiter:27764"
  arbiterDataPath: "/var/data/sessions.64"
  primaryMembers:
  - "sessionmgr03-site2:27764"
  - "sessionmgr04-site2:27764"
  secondaryMembers:
  - "sessionmgr03-site1:27764"
  - "sessionmgr04-site1:27764"
  dataPath: "/var/data/sessions.64"
```

```
  primaryMembersTag: "SITE2"
  secondaryMembersTag: "SITE1"
  siteId: "SITE2"
- title: "REPORTING-SET66"
  setName: "set66"
  oplogSize: "1024"
  arbiter: "arbiter:27766"
  arbiterDataPath: "/var/data/sessions.66"
  members:
  - "sessionmgr03-site2:27766"
  - "sessionmgr04-site2:27766"
  dataPath: "/var/data/sessions.66"
  siteId: "SITE2"
- title: "SPR-SET67"
  setName: "set67"
  oplogSize: "1024"
  arbiter: "arbiter:27767"
  arbiterDataPath: "/var/data/sessions.67"
  primaryMembers:
  - "sessionmgr01-site2:27767"
  - "sessionmgr02-site2:27767"
  secondaryMembers:
  - "sessionmgr01-site1:27767"
  - "sessionmgr02-site1:27767"
  dataPath: "/var/data/sessions.67"
  primaryMembersTag: "SITE2"
  secondaryMembersTag: "SITE1"
  siteId: "SITE2"
- title: "AUDIT-SET65"
  setName: "set65"
  oplogSize: "1024"
  arbiter: "arbiter:37017"
  arbiterDataPath: "/var/data/sessions.65"
  members:
  - "sessionmgr03-site2:37017"
  - "sessionmgr04-site2:37017"
  dataPath: "/var/data/sessions.65"
  siteId: "SITE2"
- title: "ADMIN-SET2"
  setName: "set69"
  oplogSize: "1024"
  arbiter: "arbiter:27769"
  arbiterDataPath: "/var/data/sessions.69"
  primaryMembers:
  - "sessionmgr01-site2:27769"
  - "sessionmgr02-site2:27769"
  secondaryMembers:
  - "sessionmgr01-site1:27769"
  - "sessionmgr02-site1:27769"
  dataPath: "/var/data/sessions.69"
  siteId: "SITE2"
```

Sample Mongo GR Configuration File

```
---
- title: "SESSION-SET1"
  setName: "set01"
  oplogSize: "1024"
  arbiter: "arbiter-site3:27717"
  arbiterDataPath: "/var/data/sessions.1"
  primaryMembers:
  - "sessionmgr02-site1:27717"
```

```
    - "sessionmgr01-site1:27717"
    secondaryMembers:
    - "sessionmgr02-site2:27717"
    - "sessionmgr01-site2:27717"
    dataPath: "/var/data/sessions.1/set01"
    hotStandBy: "false"
    shardCount: "4"
    seeds: "sessionmgr01:sessionmgr02:27717"
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
    siteId: "SITE1"
  - title: "SESSION-SET2"
    setName: "set07"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27722"
    arbiterDataPath: "/var/data/sessions.7"
    members:
    - "sessionmgr03-site1:27722"
    - "sessionmgr04-site1:27722"
    dataPath: "/var/data/sessions.7"
    hotStandBy: "true"
    shardCount: "4"
    seeds: "sessionmgr03:sessionmgr04:27722"
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
    siteId: "SITE1"
  - title: "BALANCE-SET1"
    setName: "set02"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27718"
    arbiterDataPath: "/var/data/sessions.2"
    primaryMembers:
    - "sessionmgr01-site1:27718"
    - "sessionmgr02-site1:27718"
    secondaryMembers:
    - "sessionmgr01-site2:27718"
    - "sessionmgr02-site2:27718"
    dataPath: "/var/data/sessions.2"
    primaryMembersTag: "SITE1"
    secondaryMembersTag: "SITE2"
    siteId: "SITE1"
  - title: "REPORTING-SET1"
    setName: "set03"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27719"
    arbiterDataPath: "/var/data/sessions.3"
    members:
    - "sessionmgr03-site1:27719"
    - "sessionmgr04-site1:27719"
    dataPath: "/var/data/sessions.3"
    siteId: "SITE1"
  - title: "SPR-SET1"
    setName: "set04"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27720"
    arbiterDataPath: "/var/data/sessions.4"
    primaryMembers:
    - "sessionmgr01-site1:27720"
    - "sessionmgr02-site1:27720"
    secondaryMembers:
    - "sessionmgr01-site2:27720"
    - "sessionmgr02-site2:27720"
    dataPath: "/var/data/sessions.4"
    primaryMembersTag: "SITE1"
```

OpenStack Sample Files - GR

```
    secondaryMembersTag: "SITE2"
    siteId: "SITE1"
  - title: "AUDIT-SET1"
    setName: "set05"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27017"
    arbiterDataPath: "/var/data/sessions.5"
    members:
    - "sessionmgr03-site1:27017"
    - "sessionmgr04-site1:27017"
    dataPath: "/var/data/sessions.5"
    siteId: "SITE1"
  - title: "ADMIN-SET1"
    setName: "set06"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27721"
    arbiterDataPath: "/var/data/sessions.6"
    primaryMembers:
    - "sessionmgr01-site1:27721"
    - "sessionmgr02-site1:27721"
    secondaryMembers:
    - "sessionmgr01-site2:27721"
    - "sessionmgr02-site2:27721"
    dataPath: "/var/data/sessions.6"
    siteId: "SITE1"
  - title: "SESSION-SET63"
    setName: "set63"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27763"
    arbiterDataPath: "/var/data/sessions.63"
    primaryMembers:
    - "sessionmgr01-site2:27763"
    - "sessionmgr02-site2:27763"
    secondaryMembers:
    - "sessionmgr01-site1:27763"
    - "sessionmgr02-site1:27763"
    dataPath: "/var/data/sessions.1/set63"
    shardCount: "4"
    seeds: "sessionmgr01:sessionmgr02:27763"
    primaryMembersTag: "SITE2"
    secondaryMembersTag: "SITE1"
    siteId: "SITE2"
  - title: "SESSION-SET68"
    setName: "set68"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27768"
    arbiterDataPath: "/var/data/sessions.68"
    members:
    - "sessionmgr03-site2:27768"
    - "sessionmgr04-site2:27768"
    dataPath: "/var/data/sessions.68"
    hotStandBy: "true"
    shardCount: "4"
    seeds: "sessionmgr01:sessionmgr02:27768"
    primaryMembersTag: "SITE2"
    secondaryMembersTag: "SITE1"
    siteId: "SITE2"
  - title: "REPORTING-SET66"
    setName: "set66"
    oplogSize: "1024"
    arbiter: "arbiter-site3:27766"
    arbiterDataPath: "/var/data/sessions.66"
    members:
    - "sessionmgr03-site2:27719"
```

```
  - "sessionmgr04-site2:27719"
  dataPath: "/var/data/sessions.66"
  siteId: "SITE2"
- title: "AUDIT-SET65"
  setName: "set65"
  oplogSize: "1024"
  arbiter: "arbiter-site3:27765"
  arbiterDataPath: "/var/data/sessions.65"
  members:
  - "sessionmgr03-site2:27017"
  - "sessionmgr04-site2:27017"
  dataPath: "/var/data/sessions.65"
  siteId: "SITE2"
```

Sample GR Cluster Configuration File - site1

```
grConfig:
  clusterInfo:
    remotePcrfclient01IP: "192.169.21.54"
    remotePcrfclient02IP: "192.169.21.55"
```

Sample GR Cluster Configuration File - site2

```
grConfig:
  clusterInfo:
    remotePcrfclient01IP: "192.169.21.19"
    remotePcrfclient02IP: "192.169.21.20"
```

Sample Set Priority File - site1

```
- op: "set-priority"
  siteId: "SITE1"
  title: "SESSION"
- op: "set-priority"
  siteId: "SITE1"
  title: "SPR"
- op: "set-priority"
  siteId: "SITE1"
  title: "BALANCE"
- op: "set-priority"
  siteId: "SITE1"
  title: "ADMIN"
```

Sample Set Priority File - site2

```
- op: "set-priority"
  siteId: "SITE2"
  title: "SESSION
```

Sample Shard Configuration File - site1

```
'- op: "modify-shards"
  setName: "set01"
  hotStandBy: "false"
  shardCount: "4"
  seeds: "sessionmgr01:sessionmgr02:27717"
- op: "modify-shards"
  setName: "set07"
  hotStandBy: "true"
  shardCount: "4"
  seeds: "sessionmgr03:sessionmgr04:27722"
```

Sample Shard Configuration File - site2

```
- op: "modify-shards"
  setName: "set63"
  hotStandBy: "false"
  shardCount: "4"
  seeds: "sessionmgr01:sessionmgr02:27763"
- op: "modify-shards"
  setName: "set68"
  hotStandBy: "true"
  shardCount: "4"
  seeds: "sessionmgr03:sessionmgr04:27768"
```

Sample Ring Configuration File

```
- op: "modify-rings"
  setName: "set01"
```

Sample Geo Site Lookup Configuration File - site1

```
grConfig:
  geoLookupConfig:
  - siteId: "SITE1"
    lookupKey:
    - "site1-gx-client.com"
```

Note The pattern matching is supported for site lookup mapping. In case the incoming host/realm does not match any of the values configured under LookupValues, request is dropped with the following exception in log:

```
GeoHASiteMappingNotFound - No realm/host to site mapping matched for:
<incoming value>
```

Sample Geo Site Lookup Configuration File - site2

```
grConfig:
  geoLookupConfig:
  - siteId: "SITE2"
    lookupKey:
    - "site2-gx-client.com"
```

Note The pattern matching is supported for site lookup mapping. In case the incoming host/realm does not match any of the values configured under LookupValues, request is dropped with the following exception in log:

```
GeoHASiteMappingNotFound - No realm/host to site mapping matched for:
<incoming value>
```

Sample Geo-tagging Configuration File - site1

```
-  op: "modify-geotag"
   title: "session"
   setName: "set01"
   primaryMembersTag: "SITE1"
   secondaryMembersTag: "SITE2"
-  op: "modify-geotag"
   title: "balance"
   setName: "set02"
   primaryMembersTag: "SITE1"
   secondaryMembersTag: "SITE2"
-  op: "modify-geotag"
   title: "spr"
   setName: "set04"
   primaryMembersTag: "SITE1"
   secondaryMembersTag: "SITE2"
```

Sample Geo-tagging Configuration File - site2

```
-  op: "modify-geotag"
   title: "session"
   setName: "set63"
   primaryMembersTag: "SITE2"
   secondaryMembersTag: "SITE1"
```

Sample Monitor Database Configuration File - site1

```
dbMonitorForLb:
  setName:
    - SPR-SET1
    - SESSION-SET1
    - BALANCE-SET1
    - ADMIN-SET1
```

```
dbMonitorForQns:
  stopUapi: "false"
  percentageSessDBFailure: 50
  setName:
    - SPR-SET1
    - SESSION-SET1
    - BALANCE-SET1
    - ADMIN-SET1
```

Sample Monitor Database Configuration File - site2

```
dbMonitorForLb:
  setName:
    - SESSION-SET63
dbMonitorForQns:
  stopUapi: "false"
  percentageSessDBFailure: 50
  setName:
    - SESSION-SET63
```