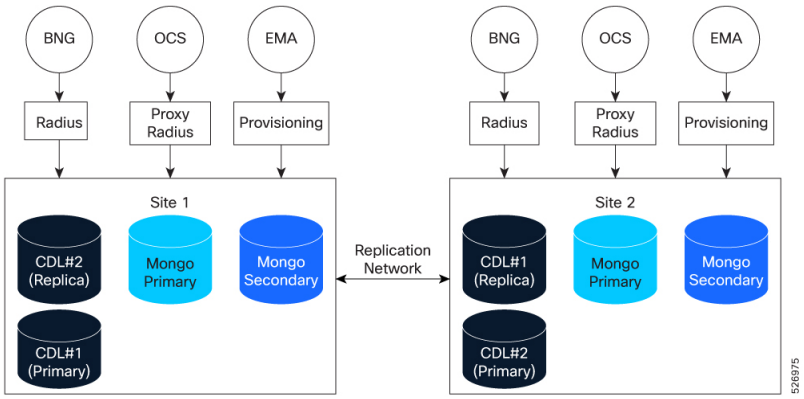
Geo-Redundancy

Geo-Redundancy

CPC can be deployed in a geographically redundant (GR) manner to maintain service availability during catastrophic failures, such as the loss of a data center. Two cnAAA clusters connect either locally or remotely. The system achieves redundancy by replicating subscriber and session data across sites.

Subscriber traffic from the BNG and OCS flows into the RADIUS endpoints at Site 1 and Site 2. Provisioning data from the EMA also flows into these endpoints. This ensures that both sites are prepared to handle active sessions and maintain service continuity during a site-level failover.

Figure 1: CPC GR deployment architecture



Redundancy is achieved by:

- Session replication: Each site operates its own CDL instance. Local session data is replicated to the remote site through the replication network.
- Subscriber replication: For subscriber management, MongoDB is used, with data persisted to disk. A single replica set can be deployed. The primary and one secondary are hosted at the local site. Two additional secondary members are hosted at the remote site. Customers may deploy additional replica

sets to meet scale requirements. Data is replicated between the local and remote sites via the replication network.

CDL session redundancy

Each site operates an individual CDL instance that stores local RADIUS sessions. This data replicates to the other site through the replication network. If a site fails, BNG traffic fails over to the remote site, which handles the traffic using replicated data.

MongoDB subscriber redundancy

MongoDB persists data to disk. A single replica set typically includes:

- Local Site: Primary and one secondary member.
- Remote Site: Two secondary members.
- Replication: In the replication process, data is transmitted through the replication network. All write operations are sent to the primary member. Read operations take place on local secondary members to reduce overhead.

MongoDB replica sets

MongoDB is a document-oriented database that stores transactional subscriber data. Data replication ensures data integrity.

- Primary node: The only member that accepts write operations and records changes in its operation log `oplog`.
- Secondary nodes: Members that replicate the primary node's `oplog` and apply operations to their data sets. These nodes handle read-only queries.
- Arbiter: A specialized member that does not store data. It participates in elections to ensure the replica set reaches a majority vote.

Election and majority logic

The system elects a new primary node only if it receives a majority of votes from the total number of voting members.

Majority formula:

```
$Majority = (V/2) + 1$ (where $V$ is the total number of voting members)
```

Example: Five-member replica set

In a configuration with five voting members (one primary, three secondaries, and one arbiter), the required majority is three votes.

Failover behavior: If the primary node fails, the two secondaries and the arbiter can still reach the majority of three votes to elect a new primary. The system uses an arbiter to maintain an odd number of voting members to prevent split-brain scenarios.

Arbiter placement limitations

Locate the arbiter at a third independent site to ensure system availability. Placing the arbiter at the same site as the primary or secondary members introduces specific limitations.

The location of the arbiter is a critical design factor. These issues occur when the arbiter is co-located with data members:

Table 1: Arbiter location limitations

Arbiter Location	Impact on System
Site 1	If Site 1 fails, the database on Site 2 cannot become the primary since it does not have the required majority vote.
Site 2	If Site 2 fails, the database on Site 1 loses its majority. It then steps down to a secondary state. System downtime occurs until manual intervention restores the database. In a split-brain scenario, the system may initiate an unnecessary role change from Site 1 to Site 2.
Site 3	This configuration ensures high availability and prevents split-brain scenarios during site-level failures. If a failure occurs at Site 1 or Site 2, the system maintains a majority vote, which ensures a successful failover to a new primary node.

CDL components

These components facilitate the storage and retrieval of session data:

- CDL endpoint pod: Provides the front-end interface to receive Create, Read, Update, and Delete (CRUD) requests from the policy engine. It uses a gRPC over HTTP2 interface to process database service requests and communicates with CDL index and slot pods.
- Slot pod: Stores session data. The CDL endpoint connects to all slot pods within the cluster to replicate session data across all pods.
- Index pod: Contains indexing data, including the primary key to slot map ID and unique secondary key to primary key mapping.

Note The system distributes index and slot pod data across maps (shards) to provide write scalability. Each map requires at least one replica for high availability. The default and recommended number of replicas for both Slot and Index is two.