Discovery Service

The Discovery Service provides an IP-based connection management service used when connecting to Redis Enterprise Software databases. When used in conjunction with Redis Enterprise Software’s other high availability features, the Discovery Service assists an application scope with topology changes such as adding, removing of nodes, node failovers and so on. It does this by providing your application with the ability to easily discover which node hosts the database endpoint. The API used for discovery service is compliant with the Redis Sentinel API.

Discovery Service is an alternative for applications that do not want to depend on DNS name resolution for their connectivity. Discovery Service and DNS based connectivity are not mutually exclusive. They can be used side by side in a given cluster where some clients can use Discovery Service based connection while others can use DNS name resolution when connecting to databases.

How discovery service works

The Discovery Service is available for querying on each node of the cluster, listening on port 8001. To employ it, your application utilizes a Redis Sentinel enabled client library to connect to the Discovery Service and request the endpoint for the given database. The Discovery Service replies with the database’s endpoint for that database. In case of a node failure, the Discovery Service is updated by the cluster manager with the new endpoint and clients unable to connect to the database endpoint due to the failover, can re-query the discovery service for the new endpoint for the database.

The Discovery Service can return either the internal or external endpoint for a database. If you query the discovery service for the endpoint of a database named “db1”, the Discovery Service returns the external endpoint information by default. If only an internal endpoint exists with no external endpoint the default behavior is to return the internal endpoint. The “@internal” is added to the end of the database name to explicitly ask for the internal endpoint. to query the internal endpoint explicitly with database name “db1”, you can pass in the database name as “db1@internal”.

If you’d like to examine the metadata returned from Redis Enterprise Software Discovery Service you can connect to port 8001 with redis-cli utility and execute “SENTINEL masters”. Following is a sample output from one of the nodes of a Redis Enterprise Software cluster:

```
$ ./redis-cli -p 8001
127.0.0.1:8001> SENTINEL masters
1) 1) "name"
2) "db1@internal"
3) "ip"
4) "10.0.0.45"
5) "port"
6) "12000"
7) "flags"
8) "master,disconnected"
9) "num-other-sentinels"
10) "0"
1) 1) "name"
2) "db1"
3) "ip"
4) "10.0.0.45"
5) "port"
6) "12000"
7) "flags"
8) "master,disconnected"
9) "num-other-sentinels"
10) "0"
```

It is important to note that, the Discovery Service is not a full implementation of the Redis Sentinel protocol. There are aspects of the protocol that are not applicable or would be duplication with existing technology in Redis Enterprise Software. The Discovery Service
implements only the parts required to provide applications with easy High Availability, be compatible with the protocol, and not rely on DNS to derive which node in the cluster to communicate with.

Note: To use Redis Sentinel, every database name must be unique across the cluster.

Redis client support

We recommend these clients that are tested for use with the Discovery Service that uses the Redis Sentinel API:

- Redis-py (Python redis client)
- HiRedis (C redis client)
- Jedis (Java redis client)
- Ioredis (NodeJS redis client)

If you need to use another client, consider using Sentinel Tunnel to discover the current Redis master with Sentinel and create a TCP tunnel between a local port on the client and the master.

Note: Redis Sentinel API can return endpoints for both master and slave endpoints. Discovery Service only supports master endpoints and does not support returning slave endpoints for a database.

Updated: January 27, 2022