Cisco Common Data Layer

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Product(s) or Functional Area	5G-PCF
Applicable Platform(s)	SMI
Feature Default Setting	Disabled – Configuration Required
Related Changes in this Release	Not Applicable
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
Added configuration support for: • Stale sessions clean-up • Remote index synchronization	2021.02.01
First introduced.	2020.01.0

Feature Description

The PCF extends support to the Geographic Redundancy (GR) version of the Cisco Common Data Layer (CDL). When the primary CDL endpoint fails, PCF attempts the same operation on the next highly rated secondary endpoint thus providing a nondisrupted N7 or Diameter message handling. If the next rated endpoint is unavailable, then PCF reattempts the operation on the subsequent endpoint that has the highest rating and so on.

For more information on the CDL concepts, see the *Ultra Cloud Core Common Data Layer Configuration Guide*.

Limitations

In the current release, this feature has the following limitations:

- The PCF attempts to reroute the calls only when it encounters gRPC errors such as UNAVAILABLE. It does not acknowledge errors that the datastore returns and actual gRPC timeouts such as DEADLINE_EXCEEDED gRPC status code.
- The PCF Engine does not resolve failures occurring with the datastore such as indexing and slot failures. The CDL layer must resolve these failures and if necessary, send an API call on the remote.

Stale Sessions Clean-Up

In the CDL sessions, PCF adds the unique session key SupiDnnKey and the pre-existing unique keys that include FramedIpv6PrefixKey. With the CDL's index overwrite detection command in the PCF Ops Center, the administrators can configure the ability to delete the old session using the same unique key while the new session is created.

The unique keys that should be used in the overwrite detection configuration are SupiDnnKey and FramedIpv6PrefixKey with the action as delete_record.

Note If two unique keys (one key mapped to the notify action and the other to the delete action) point to the same primary key, then only the notify action is considered for the primary key.

For more information on CDL components, refer to the *Cisco Common Data Layer* documentation.

Synchronizing the Index Records

Sometimes after the primary site is reinstated, the index data on both the sites may not be consistent. To reconcile the records and eliminate the discrepancy in the sites, configure the sync operation that initiates index data synchronization on the site with its remote peers.

For information on how site isolation works in PCF, see Site Isolation.

Note Configuring the sync operation may cause a negative performance impact. Cisco recommends performing this operation in a production environment that experiences a high number of inconsistent index records.

A sync operation cannot be initiated for an index instance where the remote sync is in progress.

How it Works

When you configure the CDL in PCF through the PCF Ops Center, PCF gets enabled to support multiple CDL datastore endpoints. You can configure the endpoints by specifying the IP addresses, port numbers, and assigning ratings to each endpoint. By default, PCF considers the local endpoint as the primary endpoint, which has the highest rating. PCF performs CDL API operations on the primary endpoint. If this endpoint is unavailable, then PCF routes the operations to the next highest rated endpoint. PCF keeps failing over to the accessible secondary endpoint or until all the configured secondaries are exhausted. It does not reattempt a query on the next rated endpoint if the endpoint is reachable but responds with error or timeout.

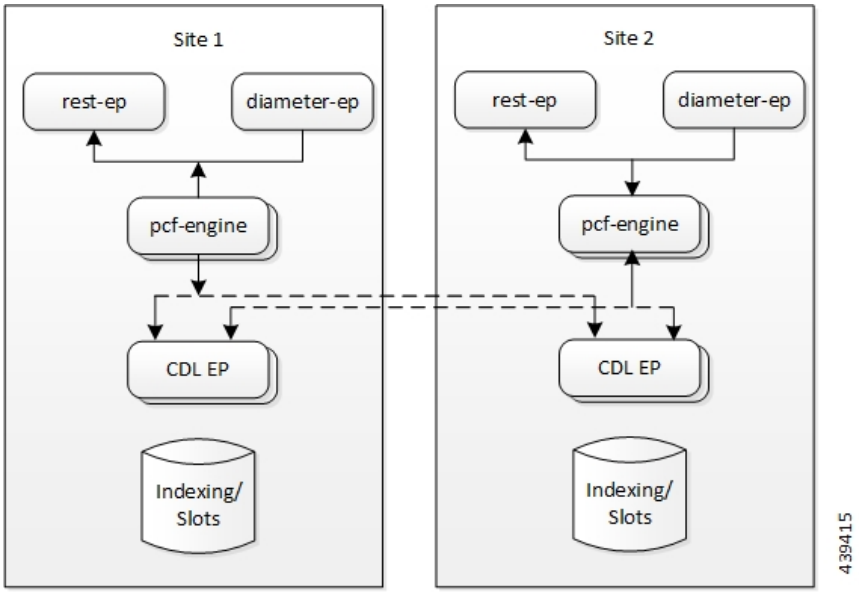
If PCF is unable to access any of the endpoints in the cluster, then CDL operation fails with the "Datastore Unavailable" error.

Architecture

You can configure CDL through PCF Ops Center. CDL in the GR mode replicates the session data across the configured sites. When PCF connects to the CDL, it always treats the local CDL endpoints as the primary endpoint and the remote endpoints as secondaries (with the appropriate rating). PCF uses the secondary endpoints when the connection to the primary endpoint fails.

The following illustration depicts the failover that happens when the PCF Engine is unable to access the primary CDL datastore endpoint.

Figure 1: CDL Datastore Architecture



Call Flows

This section describes the following call flow.

CDL Endpoint Failure Call Flow

This section describes the PCF local data store endpoint failure call flow.

Figure 2: CDL Endpoint Failure Call Flow

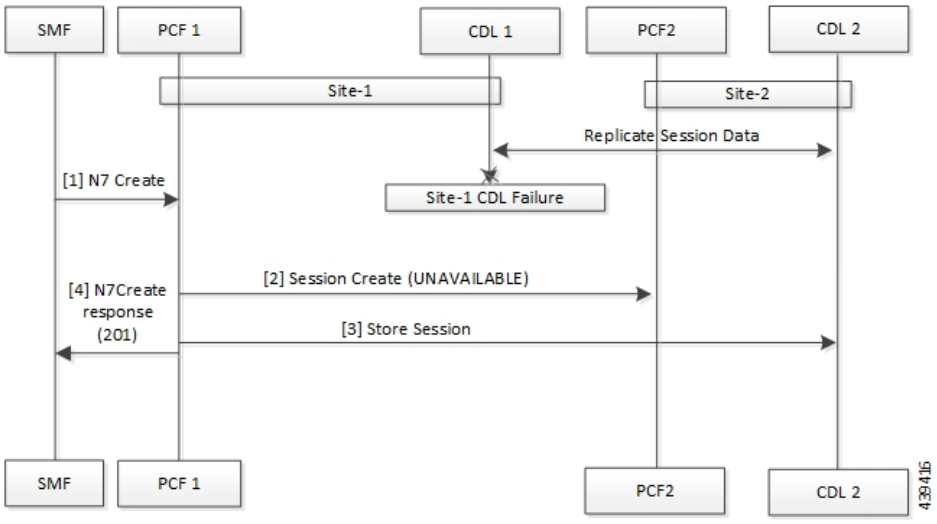


Table 3: CDL Endpoint Failure Call Flow Description

Step	Description
1	In the Site 1 environment, the SMF sends a N7 Create Request to the PCF 1 over the N7 interface.
2	The PCF 1 sends Session Create Request to the PCF 2.
3	The PCF 1 sends a Session Store Request to the CDL2.
4	The PCF 1 sends N7 Create Response to the SMF.

GR Call Flows

This section describes the possible CDL GR mode call flows scenarios that could initiate a failover to another site.

Indexing Shard Failure Call Flow

This section describes how the failover happens when two index replicas that belong to the same shard are down or unavailable.

The indexing shard failure is an example of two points-of-failure scenario where the two replicas reside on different virtual machines or hosts.

The PCF REST endpoint and PCF Engine redirect the traffic to the secondary CDL endpoint site (Site 1) based on the highest rating when the primary CDL site (Site 1) is unavailable.

Figure 3: Indexing Shard Failure Call Flow

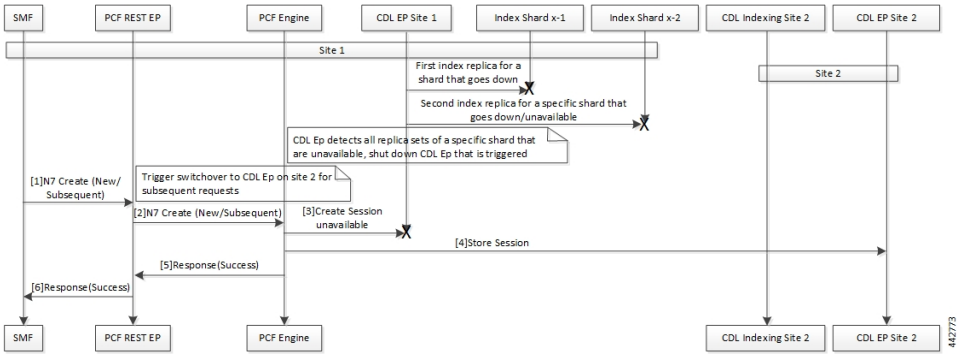


Table 4: Indexing Shard Failure Call Flow Description

Step	Description
1	In the Site 1 environment, index replica 1 and replica 2 for a configured shard has failed or unavailable. Since both the replicas for the shard are unavailable, the CDL endpoint in Site 1 is shut down and all the subsequent requests are directed to the CDL endpoint on Site 2. In the Site 1 environment, the SMF sends a Create Request to PCF REST endpoint over the N7 interface.
2	After receiving the request, the PCF REST endpoint forwards the Create Request to the PCF Engine.

Step	Description
3	The PCF Engine attempts to reach the CDL endpoint to send the Session Create Request. However, the CDL endpoint is unreachable. The PCF Engine sorts the CDL points across Site 1 and Site 2 to recognize the endpoint with the highest rating or priority.
4	The Create Request is evaluated in the stored session and the PCF Engine forwards the request to the CDL endpoint residing in Site 2.
5	After the call request is successful, the PCF Engine notifies the Success Message to the PCF REST endpoint.
6	The PCF REST endpoint forwards the Success Message to the SMF.

Slot Replica Set Failure Call Flow

This section describes how the failover happens when two slot replicas that belong to the same replica set are down or unavailable.

The slot failure is an example of two points-of-failure scenario where the two slot replicas reside on different virtual machines or hosts.

Figure 4: Slot Failure Call Flow

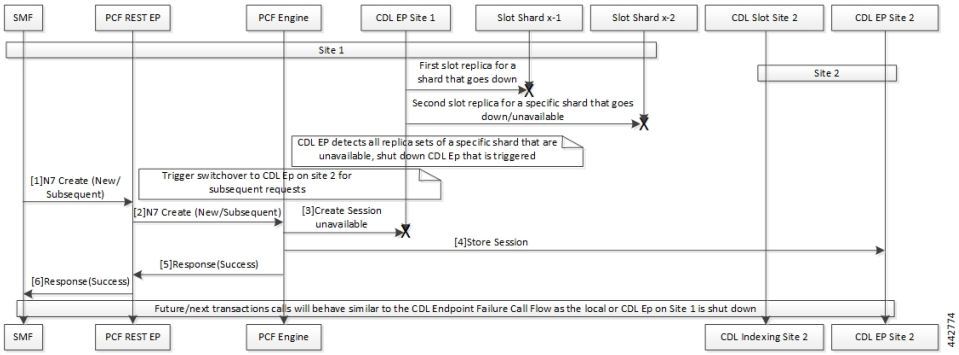


Table 5: Slot Replica Set Failure Call Flow Description

Step	Description
1	In the Site 1 environment, slot replica 1 and replica 2 for a configured shard is down or unavailable. Since both the replicas for the shard are unavailable, the CDL endpoint in Site 1 is shut down and all the subsequent requests are directed to the CDL endpoint on Site 2. In the Site 1 environment, the SMF sends a N7 Create request to PCF REST endpoint over the N7 interface.
2	The PCF REST endpoint receives the request and forwards it to the PCF Engine.
3	The PCF Engine attempts to connect the CDL endpoint to send the Session Create request. If the CDL endpoint is unreachable, the PCF Engine sorts the CDL points across Site 1 and Site 2 to recognize the endpoint with the highest rating or priority.

Step	Description
4	The Create Request is evaluated in the stored session and the PCF Engine forwards the request to the CDL endpoint residing in Site 2.
5	After the call request is successful, the PCF Engine notifies the Success message to the PCF REST endpoint.
6	The PCF REST endpoint forwards the Success message to the SMF.

Configuring Cisco Common Data Layer

This section describes how to configure the CDL endpoints.

Configuring the CDL using PCF Ops Center involves the following steps:

1. Configuring the CDL Session Database and Defining the Base Configuration
2. Configuring Kafka in CDL
3. Configuring Zookeeper in CDL

Configuring the CDL Session Database and Defining the Base Configuration

This section describes how to configure the CDL session database and define the base configuration in PCF.

To configure the CDL session database and define the base configuration in CDL, use the following configuration in the Policy Ops Center console:

```
config
  cdl
    system-id system_id
    node-type node_type
    enable-geo-replication [ true | false ]
    zookeeper replica zookeeper_replica_id
    remote-site remote_system_id
      db-endpoint host host_name
      db-endpoint port port_number
      kafka-server remote_kafka_host1 remote_port1
      kafka-server remote_kafka_host2 remote_port2
      kafka-server remote_kafka_host3 remote_port3
      exit
   cdl logging default-log-level debug_level
      cdl datastore session
      cluster-id cluster_id
      geo-remote-site remote_site_value
      endpoint replica replica_number
      endpoint external-ip ip_address
      endpoint external-port port_number
         index map map_value
         slot replica replica_slot
```

```
slot map map/shards
slot write-factor write_factor
slot notification host host_name
slot notification port port_number
slot notification limit tps
index replica index_replica
index map map/shards
index write-factor write_factor
end
```

NOTES:

- **system-id** *system_id* – This is an optional command. Specifies the system or Kubernetes cluster identity. The default value is 1.
- **node-type** *node_type* – This is an optional command. Specifies the Kubernetes node label to configure the node affinity. The default value is "session." *node_type* must be an alphabetic string of 0-64 characters.
- **enable-geo-replication [true | false]** – This is an optional command. Specifies the geo replication status as enable or disable. The default value is false.
- **zookeeper replica** *zookeeper_replica_id* – Specifies the Zooker replica server ID.
- **remote-site** *remote_system_id* – Specifies the endpoint IP address for the remote site endpoint. Configure this command only when you have set the cdl enable-geo-replication to true.
- **db-endpoint host** *host_name* – Specifies the endpoint IP address for the remote site. Configure this command only when you have set the cdl enable-geo-replication to true.
- **db-endpoint port** *port_number* – Specifies the endpoint port number for the remote site endpoint. The default port number is 8882. Configure this command only when you have set the cdl enable-geo-replication to true.
- **kafka-server** *remote_kafka_host1 remote_port1* – Specifies the Kafka server's external IP address and port number of the remote site that the remote-system-id identifies. You can configure multiple host address and port numbers per Kafka instance at the remote site. Configure this command only when you have set the cdl enable-geo-replication to true.
- **endpoint replica** *replica_number* – This is an optional command. Specifies the number of replicas to be created. The default value is 1. *replica_number* must be an integer in the range of 1 – 16.
- **endpoint external-ip** *ip_address* – This is an optional command. Lists the external IP address to expose the database endpoint. Configure this command only when you have set the cdl enable-geo-replication to true.
- **endpoint external-port** *port_number* – This is an optional command. Specifies the external port number to expose the database endpoint. Configure this command only when you have set the cdl enable-geo-replication to true. The default value is 8882.
- **slot replica** *replica_slot* – This is an optional command. Specifies the number of replicas to be created. The default value is 1. *replica_slot* must be an integer in the range of 1 – 16.
- **slot map** *map/shards* – This is an optional command. Specifies the number of partitions in a slot. The default value is 1. *map/shards* must be an integer in the range of 1 – 1024.

- **slot write-factor** *write_factor* – This is an optional command. Specifies the number of copies to be written before successful response. The default value is 1. *write_factor* must be an integer in the range of 0 – 16. Make sure that the value is lower than or equal to the number of replicas.
- **slot notification host** *host_name* – This is an optional command. Specifies the notification server hostname or IP address. The default value is datastore-notification-ep.
- **slot notification port** *port_number* – This is an optional command. Specifies the notification server port number. The default value is 8890.
- **slot notification limit** *tps* – This is an optional command. Specifies the notification limit per second. The default value is 2000.
- **index replica** *index_replica* – This is an optional command. Specifies the number of replicas to be created. The default value is 2. *index_replica* must be an integer in the range of 1 – 16.
- **index map** *map/shards* – This is an optional command. Specifies the number of partitions in a slot. The default value is 1. *map/shards* must be an integer in the range of 1 – 1024. Avoid modifying this value after deploying the CDL.
- **index write-factor** *write_factor* – This is an optional command. Specifies the number of copies to be written before successful response. The default value is 1. *write_factor* must be an integer in the range of 0 – 16.

Configuring Kafka in CDL

This section describes how to configure Kafka in CDL.

To configure the Kafka in CDL, use the following configuration:

1. Open the Policy Ops Center console and navigate to the datastore CLI.
2. To configure Kafka, use the following configuration:

```
config
   cdl kafka replica number_of_replicas
      enable-JMX-metrics [ true | false ]
      external-ip ip_address  port_number
      enable-persistence [ true | false ]
      storage storage_size
      retention-time retention_period
      retention-size retention_size
      end
```

NOTES:

All the following parameters are optional.

- **cdl kafka replica** *number_of_replicas* – Specifies the number of replicas to be created. The default value is 3. *number_of_replicas* must be an integer in the range of 1 – 16.
- **enable-JMX-metrics [true | false]** – Specifies the status of the JMX metrics. The default value is true.
- **external-ip** *ip_address port_number* – Lists the external IPs to expose to the Kafka service. Configure this command when you have set the **enable-geo-replication** parameter to true. You are required to define an external IP address and port number for each instance of the Kafka replica. For example,

if the **cdl kafka replica** parameter is set to 3, then specify three external IP addresses and port numbers.

- **enable-persistence [true | false]** – Indicates whether to enable or disable persistent storage for Kafka data. The default value is false.
- **storage** *storage_size* – Specifies the Kafka data storage size in gigabyte. The default value is 20 GB. *storage_size* must be an integer in the range of 1-64.
- **retention-time** *retention_period* – Specifies the duration (in hours) for which the data must be retained. The default value is 3. *retention_period* must be an integer in the range of 1 – 168.
- **retention-size** *retention_size* – Specifies the data retention size in megabyte. The default value is 5120 MB.

Configuring Zookeeper in CDL

This section describes how to configure Zookeeper in CDL.

To configure Zookeeper in CDL, use the following configuration:

1. Open the Policy Ops Center console and navigate to the datastore CLI.
2. To configure the parameters, use the following configuration:

```
config
  cdl zookeeper data-storage-size data_storage
    log-storage-size log_storage
    replica number_of_replicas
    enable-JMX-metrics [ true | false ]
    enable-persistence [ true | false ]
    end
```

NOTES:

All the following parameters are optional.

- **cdl zookeeper data-storage-size** *data_storage* – Specifies the size of the Zookeeper data storage in gigabyte. The default value is 20 GB. *data_storage* must be an integer in the range of 1-64.
- **log-storage-size** *log_storage* – Specifies the size of the Zookeeper data log's storage in gigabyte. The default value is 20 GB. *log_storage* must be an integer in the range of 1-64.
- **replica** *number_replicas* – Specifies the number of replicas that must be created. The default value is 3. *number_replicas* must be an integer in the range of 1-16.
- **enable-JMX-metrics [true | false]** – Specifies the status of the JMX metrics. The default value is true.
- **enable-persistence [true | false]** – Specifies the status of the persistent storage for Zookeeper data. The default value is false.

Sample Configuration

The following is a sample configuration of CDL in the HA environment.

```
cdl system-id   system_i
cdl enable-geo-replication true
cdl zookeeper replica num_zk_replica
cdl datastore session
 endpoint replica ep_replica
index map index_shard_count
 slot replica slot_replica
 slot map slot_shard_count
exit
cdl kafka replica kafka_replica
```

Configuring the CDL Endpoints

This section describes how to configure the CDL endpoints.

Configuring the CDL endpoints involves the following steps:

1. Configuring the External Services
2. Associating the Datastore with the CDL Endpoint Service

Configuring the External Services

This section describes how to configure the external services in PCF.

CDL gets deployed in the GR environment as part of the SMI deployment procedure. By default, the CDL endpoints are available in the Datastore CLI node of the PCF Ops Center. However, you are required to configure these endpoints.

For each CDL site and instance, configure external service with the IP address and port number that corresponds to the site and instance.

1. Open the Policy Ops Center console and navigate to the datastore CLI.
2. To configure the parameters, use the following configuration:

```
config
    external-services site_name
    ips ip_address
    ports port_number
    end
```

NOTES:

- **external-services** *site_name* – Specifies the CDL site or instance name.
- **ips** *ip_address* – Specifies the IP address on which the CDL endpoint is exposed.
- **ports** *port_number* – Specifies the port number on which the CDL endpoint is exposed.

Associating the Datastore with the CDL Endpoint Service

This section describes how to configure the external service for each CDL endpoint service that you plan to use.

To configure the external service for each CDL endpoint service, use the following configuration:

1. Open the Policy Ops Center console and navigate to the datastore CLI.
2. To associate the datastore with CDL endpoint service, use the following configuration:

```
config
  datastore external-endpoints service_name
  port port_number
  rating rating_priority
  end
```

NOTES:

- **datastore external-endpoints** *service_name* – Specifies the service name that belongs to the external services.
- **port** *port_number* – Specifies the port number where the external service resides.
- **rating** *rating_priority* – Specifies the rating or priority of the external service. PCF gives preference to the endpoints with the higher ratings.

Starting the Remote Index Synchronization

This section describes how to initiate the remote index synchronization.

Before configuring the remote index sync, ensure that the geo-remote-site parameter for CDL is configured.

To start the remote index synchronization, use the following configuration:

```
cdl
  actions
    remote-index-sync start [ map-id map_id | slice-name slice_name ]
    end
```

NOTES:

- **cdl** – Enters the CDL configuration mode.
- **remote-index-sync start** – Configures the remote index sync feature.
- **map-id** *map_id* – Specifies the index mapID for which the remote index sync procedure should start. By default, remote index sync is initiated for all the index instances.

  Using this parameter you can specify a maximum of 5 mapIDs.
- **slice-name** *slice_name* – Specifies the slice name for which the remote index sync procedure should start. By default, remote index sync is initiated for all the sliceNames. There is no limit to the number of sliceNames that you can specify.

Sample Configuration

```
cdl actions remote-index-sync start map-id { 1 } map-id { 2
} slice-name { session-1 } slice-name { session-2 }
```

Viewing the Remote Index Synchronization Status

This section describes how to view the status of the index synchronization procedure that you have executed.

To view the status of the index sync procedure, use the following configuration:

```
cdl
  actions
    remote-index-sync status
    end
```

NOTES:

- **remote-index-sync status** – Displays the status of the index instances for which the syncing with the remote peers is in progress.

Sample Output

```
syncing-instances 'index-mapID-1-instanceID-1, index-mapID-
1-instanceID-2, index-mapID-2-instanceID-1, index-mapID-2-
instanceID-2'
```

Configuring the Stale Sessions Clean-Up Support

This section describes how to clean up the sessions that are inactive.

To clean up the stale sessions, use the following configuration:

```
config
  cdl
    datastore session datastore_name
      features
             index-overwrite-detection [ max-tps | queue-size |
unique-keys-prefix ]
             action [ notify-record | log-record | delete-record ]
             exit
              exit
```

Sample Configuration

```
cdl datastore session
features index-overwrite-detection unique-keys-prefix SupiDnnKey
action delete-record
exit
features index-overwrite-detection unique-keys-prefix FramedIpv6PrefixKey
action delete-record
exit
exit
```

NOTES:

- **cdl**– Enters the CDL configuration mode.

- **datastore session** *datastore_name* – Specifies the CDL datastore session.
- **index-overwrite-detection [max-tps | queue-size | unique-keys-prefix]** – Configures the index keys overwrite detection capability. The parameter has the following subparameters:
  - **max-tps** – Configures the TPS per cdl-endpoint at which the stale record notification is sent. This parameter is applicable only when the action is "notify-record". The accepted value range is 1..2000. The default value is 200.
  - **queue-size** – Configures the queue size for each cdl-endpoint. The default value is 1000.
  - **unique-keys-prefix** – Configures the list of uniqueKey prefixes for the index overwrite detection and the action that must be performed on successful detection. Cisco recommends specifying the uniqueKey prefixes as SupiDnnKey and Ipv6Prefix.
- **action [notify-record | log-record | delete-record]** – Specifies the action that must be taken on detecting a stale record. PCF supports only the delete-record action, which cleans up the duplicate sessions.

Note If configuring the stale session clean-up feature for the first time on your system, Cisco recommends performing the configuration after both the GR sites are upgraded to the same software version.

Important The delete-record action on key SupiDnnKey command takes effect only when the required key SupiDnnKey is added in the CDL sessions.

Troubleshooting Information

To view the status of the clean up status of the stale sessions, review the warning logs in index pods.

You can review the logs to debug the stale sessions issues by setting the index.overwrite.session log to INFO level.

Example:

```
cdl logging logger index.overwrite.session
level info
exit
```

Stale Sessions Clean-Up OA&M Support

This section describes operations, administration, and maintenance information for this feature.

Statistics

This section provides the list of counters that are generated for scenarios where the stale session clean-up process is initiated.

The following metrics track the counter information:

- overwritten_index_records_deleted: Captures the total number of records deleted due to overwritten or duplicate unique keys at index

  Sample query: overwritten_index_records_deleted

  The following labels are defined for this metric:

  - errorCode: The error code in the DB response. Example: 0, 502.
  - sliceName: The name of the logical sliceName. Example: session

- overwritten_index_records_skipped: Captures the total number of records detected as stale, but dropped when the queue is full while processing the records for notify or delete.

  Sample query: overwritten_index_records_skipped

  The following labels are defined for this metric:

  - action: The action that was supposed to be performed for the stale record. Example: delete, notify.
  - sliceName: The name of the logical sliceName. Example: session

For information on statistics, see *Ultra Cloud Core Common Data Layer Configuration and Administration Guide*.