UCC CDL Release Change Reference, Release 1.10

Features and Behavior Changes Quick Reference

The following table indicates the default values of features and behavior changes in this release.

Features/ Behavior Changes	Default	Release Introduced/ Modified
CDL Rack Conversion, on page 1	Enabled – Always-on	1.10.2
CDL Zone Upgrade, on page 2	Enabled – Always-on	1.10.0
Remote Site Monitoring, on page 5	Enabled – Always-on	1.10.0

CDL Rack Conversion

Feature Summary and Revision History

Summary Data

Applicable Product (s) or Functional Area	KVM-based application deployment support
Applicable Platforms	Bare Metal, OpenStack, VMware
Feature Default Setting	Enabled – Always-on
Related Changes in this Release	Not Applicable

Related Documentation	*UCC CDL Configuration and Administration Guide, Release 1.10*

Revision History

Revision Details	Release
First introduced.	CDL 1.10.2

Feature Description

CDL supports rack conversion from half height to full height without draining traffic using the rebalancing method.

The rebalancing method does not affect the existing and new calls during migration. This feature addresses the index map rebalancing for CDL.

For more information, refer to the *UCC CDL Configuration and Administration Guide, Release 1.10*.

Troubleshooting

To view the rebalancing logs, configure the following logger in CLI:

```
cdl logging logger datastore.idx-rebalance.session
level warn
exit
```

The following metric is introduced in the index pod to know the total number of index keys that are rebalanced.

Metric	Label
index_rebalanced_keys_total	cdl_slice, shardId

CDL Zone Upgrade

Feature Summary and Revision History

Summary Data

Applicable Product (s) or Functional Area	KVM-based application deployment support
Applicable Platforms	Bare Metal, OpenStack, VMware
Feature Default Setting	Enabled – Always-on
Related Changes in this Release	Not Applicable

Related Documentation	*UCC CDL Configuration and Administration Guide, Release 1.10*

Revision History

Revision Details	Release
First introduced.	CDL 1.10.0

Feature Description

CNDP supports zone-based upgrades where all the nodes in an upgrade zone are upgraded in parallel. This avoids the time that is taken to upgrade large clusters. CDL utilizes the K8s Topology Spread Constraints feature to spread the pods across upgrade zones.

To support this in-service for CDL, the pods must be aware of upgrade zones and ensure that all replicas for a slot, index, or Kafka do not get scheduled in the same upgrade zone. For example, if both slot replicas of the same map are in the same upgrade zone, both replicas go down at the same time and cause session loss in a non-GR scenario. The CDL endpoint stops serving requests until one replica is back.

Ensure that at least three upgrade zones are present. Any node failure can result in Pending state for pods even if another zone has space to schedule the pod.

There is no explicit configuration that is required in CDL or NF Ops Center to enable this feature. The cluster deployer itself passes the configuration.

Topology Spread Constraint

CDL uses the K8s topology spread constraints feature to spread pods across upgrade zones so that if an upgrade zone is down, all the redundant pods are not lost at the same time.

The following fields are defined for the topology spread constraint:

- **maxSkew**—Defines the maximum permitted difference between the number of matching pods in the target topology and the global minimum (the minimum number of pods that match the label selector in a topology domain). For example, if you have 3 zones with 2, 4, and 5 matching pods respectively, then the global minimum is 2 and maxSkew is compared relative to that number.
- **whenUnsatisfiable**—Indicates how to deal with a pod if it does not satisfy the spread constraint:
    - **DoNotSchedule** (default) indicates the scheduler not to schedule the pod.
    - **ScheduleAnyway** indicates the scheduler to schedule the pod while prioritizing nodes that minimize the skew.
- **topologyKey** is the key of node labels. If two nodes are labelled with this key and have identical values for that label, the scheduler treats both nodes as being in the same topology. The scheduler tries to place a balanced number of pods into each topology domain.
- **labelSelector** is used to find matching pods. Pods that match this label selector are counted to determine the number of pods in their corresponding topology domain.

For CDL pods, the following is an example of the fields defined for topology spread constraint:

- **topologyKey: smi.cisco.com/upgrade-zone**
- **maxSkew: 1**
- **whenUnsatisfiable: DoNotSchedule**
- **labelSelector:** Selects all pods matching the deployment or stateful set

Based on this constraint, the K8s scheduler attempts to distribute the pods evenly in a deployment or stateful set across zones with maxSkew of 1.

- For slot or index pods that have two replicas, then the two replicas are always in separate upgrade zones.
- For cdl-ep or Kafka pods, the pods will be distributed almost evenly with a maxSkew of 1.
- If the pod is unable to satisfy the **maxSkew: 1** constraint, the pod will remain in Pending state as the **whenUnsatisfiable** field is set to **DoNotSchedule**. This constraint does not allow the maxSkew to be violated.

Note The topology spread constraints are only added for cdl-ep, cdl-slot, index, Kafka, or stateful sets deployment. There is no change for etcd and zookeeper as they are scheduled on primary or OAM nodes which will be upgraded one by one.

Limitations and Restrictions

This section describes the known limitations and restrictions for CDL zone upgrade:

- CDL supports NF deployments with Kafka replica 2 or Kafka replica 3. For deployments with Kafka replica 3, CDL defines at least three upgrade zones. If there are three replicas and two upgrade zones, then one upgrade zone will have two replicas and the other will have one replica. In this scenario where the upgrade zone having two replicas goes down, only one replica will be available to serve traffic.
- If the nodes in an upgrade zone are fully utilized and if one or more nodes in that zone go down, the pods on one or more nodes that went down may be in Pending state.

  The following conditions apply for same zone nodes and different zone nodes:

  - Same zone nodes—Nodes in the same zone do not have the requested CPU or memory resources for the pod to be scheduled on it.
  - Different zone nodes—One of the following conditions applies for different zone nodes:
    - Nodes in another zone do not have the requested CPU or memory resources for the pod to be scheduled on it.
    - Nodes in another zone have the requested CPU or memory resources for the pod to be scheduled. With scheduling, the pod will violate the maxSkew:1 constraint.

Remote Site Monitoring

Feature Summary and Revision History

Summary Data

Applicable Product (s) or Functional Area	KVM-based application deployment support
Applicable Platforms	Bare Metal, OpenStack, VMware
Feature Default Setting	Enabled – Always-on
Related Changes in this Release	Not Applicable
Related Documentation	*UCC CDL Configuration and Administration Guide, Release 1.10*

Revision History

Revision Details	**Release**
First introduced.	CDL 1.10.0

Feature Description

CDL endpoint monitors the remote site connection (replication connection and internal operational connection) using the CDL Ping RPC every 30 seconds. If ping fails three times for any of the connections, then recreate that connection and close the old connection.

Remote site monitoring is configurable and it is enabled by default. Use the **cdl datastore session features remote-site-connection-monitoring enable** CLI command to enable or disable remote site connections.

Configuring Remote Site Connection

To enable or disable remote site connection, use the following CLI command:

```
config
  cdl datastore session features remote-site-connection-monitoring enable
 [ true | false ]
   exit
```

Troubleshooting Information

To view the logs for remote site connection on an endpoint, use the following configuration:

```
cdl logging logger ep.remoteConnection.session
level trace
exit
```