Cisco Common Data Layer

概要

Cisco Common Data Layer（CDL）は、クラウドネイティブなすべてのアプリケーションに対応する、高性能の次世代キーバリュー（KV）データストアレイヤです。これらのアプリケーションは、高可用性（HA）と Geo HA 機能を備えた状態管理として CDL を使用します。CDL は以下を提供します。

- AMF、SMF、PCF などのさまざまなネットワーク機能（NF）にまたがる共通のデータストアレイヤ：マイクロサービス。
- 低遅延の読み取りと書き込みを実現するマルチプライマリサポート。
- 純粋なインメモリストレージ。
- セッション関連のタイマーを実行し、タイマーが切れた時に NF に通知します。
- 高速フェールオーバーによる高可用性と Geo 冗長性

マニュアルの変更履歴

表 *1:* マニュアルの変更履歴

改訂の詳細	リリース
• CDL は古いインデックスレコードについて、識別と必要なアクションの実行をサポートします。 • ピアサイトが隔離サイトの GR フェールオーバー通知を処理できるようになっています。 • CDL は、インデックスをリモートピアと同期させるためのユーティリティを提供します。 • Grafana ダッシュボードの拡張により、GR 接続ステータスが表示されます。	2020.02.3

アーキテクチャ

CDL は次のモードで展開できます。

- **HA のみ**
- **Geo HA**

（注） HA のみのデプロイメントでは、ローカルサイトでブレード障害に対する冗長性が提供されます。ただし、完全なサイトまたは K8s クラスタに到達できない場合、HA のみのデプロイメントではセッション冗長性は提供されません。

次の図は、CDL 導入モデルのハイレベルアーキテクチャを示しています。

図 *1 : CDL* 導入モデル

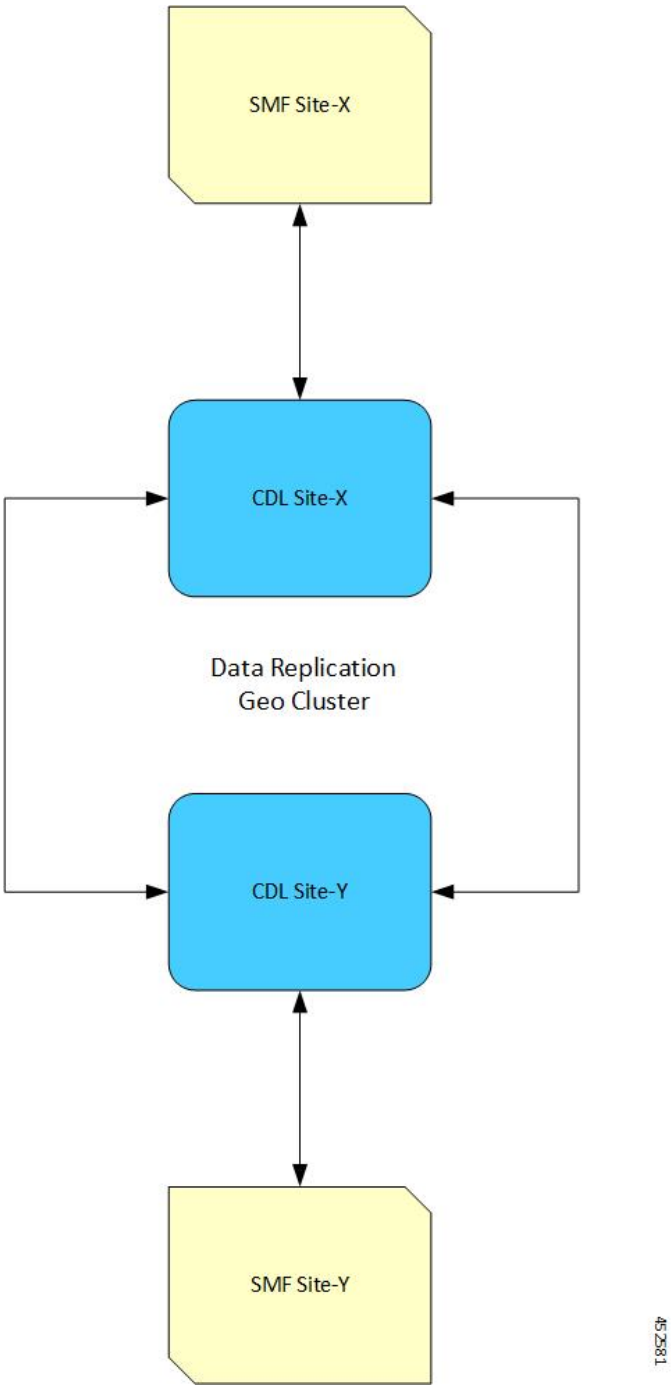
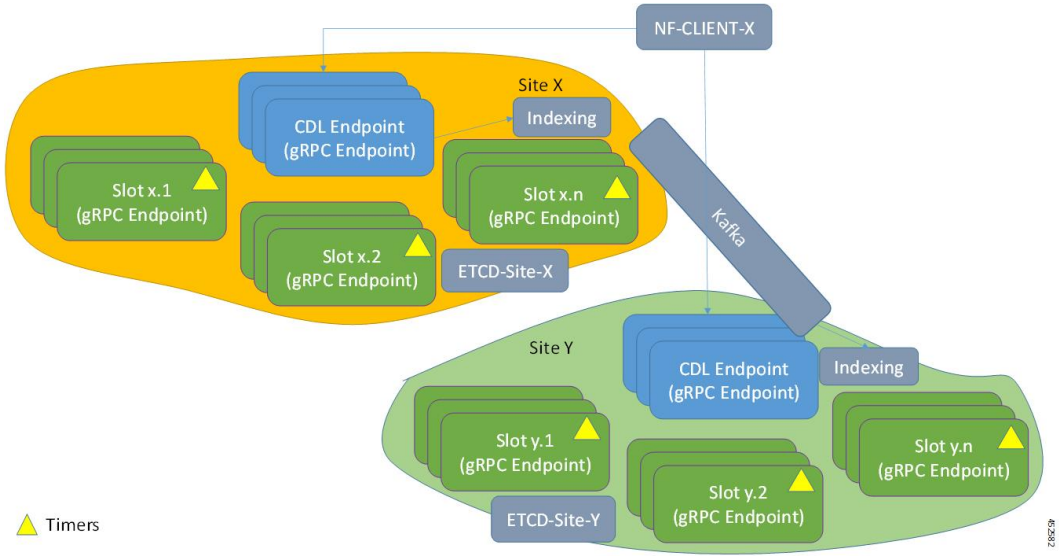
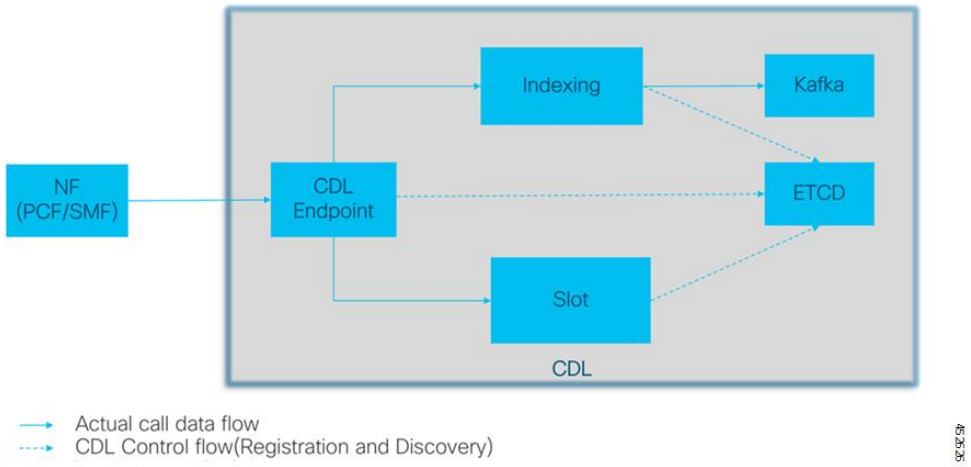

SMF Site-X
CDL Site-X
Data Replication
Geo Cluster
CDL Site-Y
SMF Site-Y
452581

図 *2 : CDL* マイクロサービス



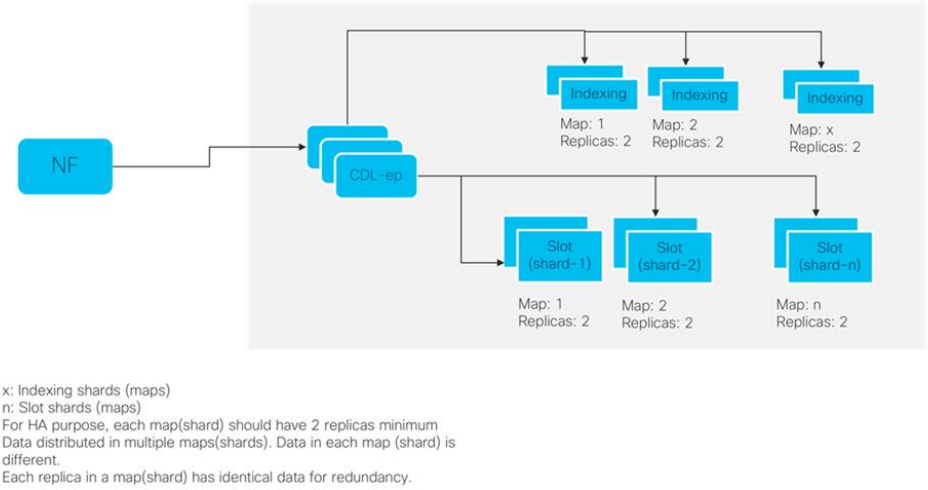
次の図は、CDL のハイレベルアーキテクチャを示しています。

図 *3 : CDL* のアーキテクチャ



次の図は、主要なコンポーネントを含む CDL のハイレベルアーキテクチャを示しています。

図 *4 :* 主要なコンポーネントを含む *CDL* アーキテクチャ



CDL エンドポイント

CDL エンドポイントポッドは、NF アプリケーションから CDL に対する要求を受信するためのフロントエンドです。CDL エンドポイントは、データベースサービス要求を処理するために、NF クライアントに向けて HTTP2 インターフェイスを介して gRPC を公開します。各ポッドは、次の属性で始まります。

- **systemID**：このパラメータは、サイト ID（例：Site）を指定します。
- **clusterID**：このパラメータは、サイト内の一意のデータストア ID（例：X、Y）を指定します。

CDL エンドポイントは、NF アプリケーションから作成、読み取り、更新、削除（CRUD）要求を受信します。これらの操作を実行するために、GRPC を介して CDL インデックスおよびスロットポッドと通信します。

CDL エンドポイントは、サイト内の他のポッドについて、*etcd*を使用して自動的に学習します。また、Cisco Data Store K8s ポッドは本質的にステートレスであり、セッションのスティッキ性は維持されません。CDL エンドポイントは NF アプリケーションから要求を受信し、GRPC を介して内部的にスロットポッドおよびインデックスポッドと通信して、それに応じて要求を処理します。処理が完了すると応答を返します。

CDL エンドポイントが作成、削除、または更新セッションを受信すると、受信コンテナはそれをスロットポッドおよびインデックスポッドにレプリケートし、「n」個の書き込みがインデックスおよびスロットのマイクロサービスによって確認（ACKed）された場合にのみ、応答を送信します。

（注） 実稼働環境でCDLエンドポイントポッドを展開するには、最大4つの仮想CPU（vCPU）が必要です。

単一のコンピューティングまたはノード障害

単一のコンピューティングまたはノードで障害が発生すると、*cdl-ep*ポッドの1つがダウンし、他のセッションノードでスケジュールを試行します。他のすべてのセッションノードに *cdl-ep* がすでにある場合、*cdl-ep* ポッドにノードのアンチアフィニティが定義されているため、ポッドのスケジュール設定は行われません。したがって、単一のノード障害によって残りの *cdl-ep* ポッドがトラフィックを処理することになり、多くの要求による負荷がかかる可能性があります。ノード障害から回復した後、以前は保留状態にあった*cdl-ep*ポッドが、回復したノードで要求を処理するためのスケジュール設定を開始します。

複数のコンピューティングまたはノード障害

すべてのセッションノードに障害が発生すると、それに合わせてすべての*cdl-ep*ポッドがダウンします。Geo 高可用性（GeoHA）が設定されている場合、NF アプリケーションはリモートサイトと通信し、ローカルサイトでセッションの処理を続行します。ただし、高可用性（HA）シナリオでは、これにより完全なダウンタイムが発生します。

スロット

CDL スロットポッドには、実際のセッションデータが保存されます。CDL エンドポイントはクラスタ内のすべてのスロットポッドに接続し、セッションデータをすべてのポッドにレプリケートします。これらのマイクロサービスは、内部 gRPC インターフェイスを Cisco Data Store に公開するために展開された K8s ポッドです。各ポッドは、次の属性で始まります。

- **systemID**：このパラメータは、サイト ID（例：Site-1）を指定します。
- **clusterID**：このパラメータは、サイト内の一意のデータストア ID（例：Session）を指定します。
- **mapID**：このパラメータは、クラスタ内のレプリカセットIDを指定します。（例：map-1、map-2、..map-n）。
- **instanceID**：このパラメータは、レプリカセット内のインスタンスIDを指定します。（例：map-1.instance-1、map-1.instance-2）

各スロットポッドは、限られた数のセッションと、保存されているセッションデータの事前に割り当てられたメモリを保持します。また、レプリカセット（mapID）内の各レプリカには、定義されたアンチアフィニティルールがあり、同じブレードフォームが同じレプリカセットの複数のメンバーまたはインスタンスをホストするのを防ぎます（ブレードまたはノードに障害が発生した場合の高可用性のため）。各スロットポッドは、タイマーと最後に更新された TS を維持します。スロットポッドは、競合が検出された場合、タイマーが切れたときにアクションを実行するために、クライアント NF への通知コールバックを生成します。

（注） 実稼働環境でスロットポッドを展開するには、最大で 2 つの vCPU が必要です。

ポッドのフェールオーバーと回復が発生した場合、スロットポッドは以下から回復します。

- **ローカルレプリカメンバー**：スロットは gRPC ストリームから一括で直接読み取り、データを回復します。
- **リモートレプリカメンバー**：同期に使用できるローカルレプリカがない場合、スロットは同じマップのリモートサイトインスタンスからデータを読み取ります。

次の図は、ローカルピアおよびリモートピアからのスロットリカバリプロセスを示しています。

図 *5:* ローカルピアからのスロットリカバリ

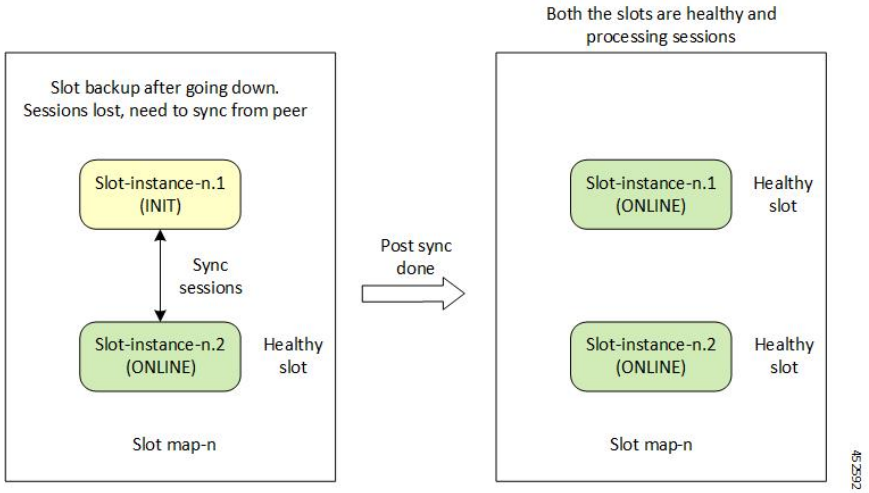
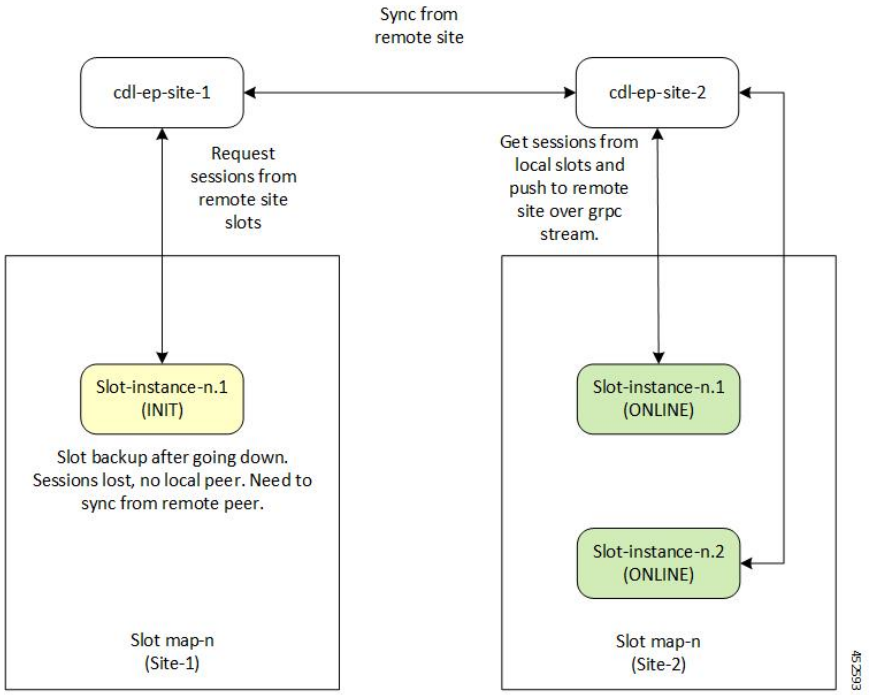


図 *6:* リモートピアからのスロットリカバリ



データのスライシング

データのスライシングでは、CDL がスライスとして論理的に分割され、ネットワーク機能（NF）から受信したスライス名に基づいてセッションデータが保存されます。

データのスライシングを使用すると、1 つ以上の NF がさまざまなタイプのセッションデータを CDL の専用スライスに保存できます。

スライス名が設定されていない場合は、**session** というデフォルトのスライス名が使用されます。

設定は次のとおりです。

```
cdl datastore <datastore name> slice-names [ <sliceName 1> <sliceName 2> ... <sliceName n> ]
```

サンプル設定を次に示します。

```
cdl datastore session slice-names [ session1 session2 ]
```

（注）
- スライス名が NF の ops-center または CDL の ops-center で設定されている場合、NF からのすべての要求に有効なスライス名が必要です。スライス名が設定されているものと違うか空の場合、エラーコードが表示され要求は拒否されます。
- スライス名が設定されていない場合、NF 要求はデフォルトの **session** にルーティングされます。
- デプロイメント後の実行中のシステムでは、スライス名を更新できません。

CDL スロットデータの削除

特定のシナリオでは、CDL レコードがスロットでは見つかりますが、インデックスポッドでは見つかりません。そのようなレコードに関するスロットからアプリケーションへの通知では、値が正しく受信されません。インデックスデータが削除されない場合、スロット内のレコードは削除されません。

CDL スロットデータを削除する前に、次のことを確認してください。

- アプリケーションへの通知数がしきい値（デフォルト値の 3）を超えた場合、レコードが古い状態であると考えられます。
- これにより、いずれかのインデックスポッド（ローカルまたはGeo リモートサイト）で対応するレコードを見つけるために、検証チェックがトリガーされます。
- インデックスのマップ ID に不一致がある場合、またはマップ ID がすべてのインデックスポッドで見つからない場合、クリーンアップが呼び出され、ローカルサイトとリモートサイトのレコードが削除されます。

古いレコードを削除するために、次のパラメータが導入されました。

disable-auto-deletion：true に設定すると、古い CDL レコードは削除されません。古いレコードの自動削除は、デフォルトで有効になっています。

notification-retry-count：アプリケーションから更新を受信せずにアプリケーションに送信するタイマー切れ通知の最小再試行回数を指定します。notification-retry-count の回数を超えても更新を受信しない場合、CDL はスロットレコードが古いかどうかのチェックに進みます。デフォルトの回数は 3 です。

CDL のサンプル設定は次のとおりです。

古いスロットレコードの自動削除機能を無効にするには、次の手順を実行します。

```
cdl datastore session
features slot-stale-session-detection disable-auto-deletion true
exit
```

notification-retry-count は、5 などの新しい値に変更できます。これは、タイマー切れ通知を 5 回試行した後、データが古いかどうかのチェックに進むことを示しています。

```
cdl datastore session
features slot-stale-session-detection notification-retry-count 5
exit
```

トラブルシューティング

エンドポイントおよびスロットポッドで古い CDL スロットデータの障害対応ログを有効にするには、次の設定を使用します。

```
cdl logging logger ep.staleRecord.session
level info
exit

cdl logging logger slot.staleRecord.session
level info
exit
```

単一のコンピューティングまたはノード障害

単一のコンピューティング障害が発生した場合、CDL は一部のスロットおよびインデックスポッドの 1 つ少ない *cdl-ep* ポッドとレプリカで機能し続けます。単一のレプリカにより、CDL スロットマップはすべての要求の処理を続行します。サービスを停止したノードがサービスに戻ると、「ローカルピアからのスロットリカバリ」の図に示されているように、ノードのスロットポッドがそれぞれのピアと同期します。

複数のコンピューティングまたはノード障害

複数のコンピューティング障害が発生した場合、スロットマップのすべてのレプリカがダウンします。たとえば、2 つのノードがダウンしている場合、スロットマップの両方のレプリカがダウンします（スロットレプリカ数のデフォルト値は 2）。これにより、ローカルのレプリカセット全体が機能不全となり、すべての *cdl-ep* ポッドのシャットダウンがトリガーされます。また、GeoHA が設定されている場合は、CDL Geo レプリケーション（GR）がトリガーされます。GeoHA が設定されている場合、NF アプリケーションはリモートサイトの *cdl-ep* と通信して要求を処理します。GeoHA 設定がない場合、最終的にサービスのダウンタイムが発生します。

障害が発生したノードが回復すると、スロットポッドとスロットレプリカセットが回復し、リモートサイトから同期されます（障害発生中に両方のレプリカがダウンした場合）。これは、GeoHA が設定されている場合にのみ発生します。リモートピアからのスロットリカバリの図は、ローカルのレプリカセット全体がダウンした場合のリモートピアからの最初の同期を示しています。

インデックス作成

インデックス作成コンテナには、インデックス作成データが含まれます。インデックスポッドには、次の 2 つの重要な情報が保存されます。

- スロットマップ ID のプライマリキー。
- セカンダリキーからプライマリキーへの固有のマッピング。

インデックスエントリは、セッション接続時に作成され（使用例の場合）、切断時に削除されます。インデックス作成では、インメモリ KV ストアを使用してキーをメモリに保存し、マルチプライマリ書き込みを提供します。インデックス作成書き込みログ（設定操作と削除操作）も、リモートサイトのレプリケーション用に Kafka ポッドに書き込まれます。インデックスポッドの 1 つが Kafka でログを書き込み、リモートサイトでそのログを受信します。対応する操作もリモートサイトで実行されます。

（注） 実稼働環境でインデックスポッドを展開するには、最大で 2 つの vCPU が必要です。

ポッドのフェールオーバーと回復が発生した場合、インデックスポッドは以下から回復します。

- **ローカルレプリカメンバー**：インデックスは gRPC ストリームから一括で直接読み取り、データを回復します。
- **リモートレプリカメンバー**：同期に使用できるローカルレプリカがない場合、インデックスは同じマップのリモートサイトインスタンスからデータを読み取ります。

次の図は、ローカルピアおよびリモートピアからのインデックス リカバリ プロセスを示しています。

図 7: ローカルピアからのインデックスリカバリ

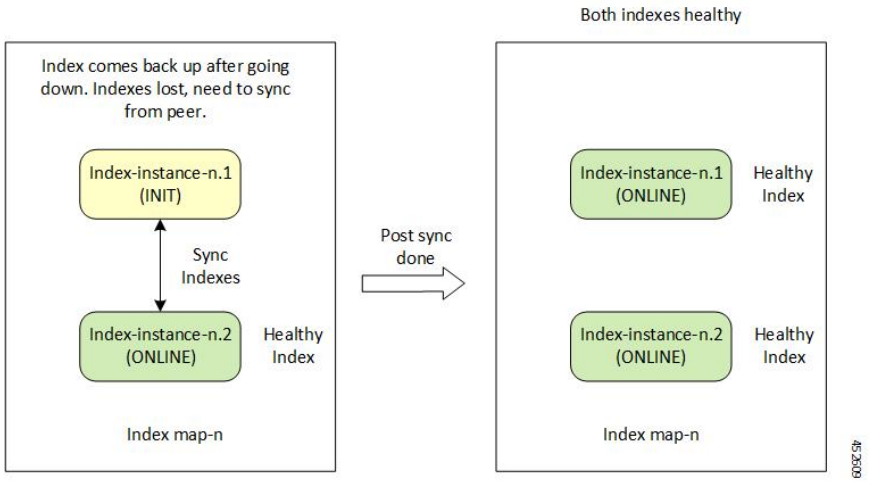
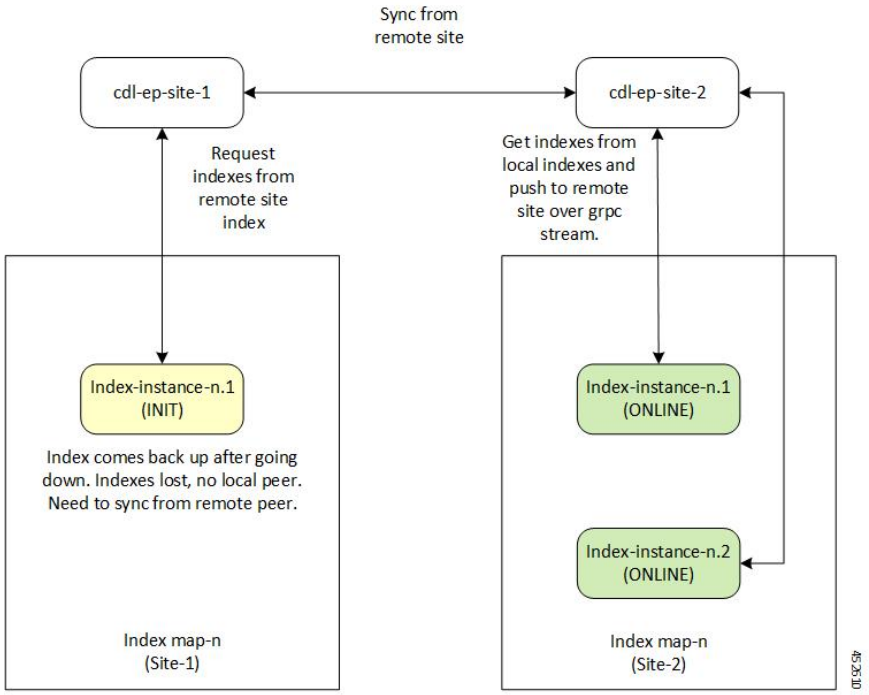


図 *8*: リモートピアからのインデックスリカバリ



単一のコンピューティングまたはノード障害

単一のコンピューティング障害が発生した場合、CDL は一部のスロットおよびインデックスポッドの 1 つ少ない *cdl-ep* ポッドとレプリカで機能し続けます。単一のレプリカにより、CDL スロットマップはすべての要求の処理を続行します。サービスを停止したノードがサービスに戻ると、「ローカルピアからのインデックスリカバリ」の図に示されているように、ノードのインデックスポッドがそれぞれのピアと同期します。

複数のコンピューティング障害

複数のコンピューティング障害が発生した場合、スロットマップのすべてのレプリカがダウンします。たとえば、2 つのノードがダウンしている場合、スロットマップの両方のレプリカがダウンします（スロットレプリカ数のデフォルト値は 2）。これにより、ローカルのレプリカセット全体が機能不全となり、すべての *cdl-ep* ポッドのシャットダウンがトリガーされます。また、GeoHA が設定されている場合は、CDL Geo レプリケーション（GR）がトリガーされます。GeoHA が設定されている場合、NF アプリケーションはリモートサイトの *cdl-ep* と通信して要求を処理します。GeoHA 設定がない場合、最終的にサービスのダウンタイムが発生します。

障害が発生したノードが回復すると、インデックスポッドとインデックスレプリカセットが回復し、リモートサイトから同期されます（障害発生中に両方のレプリカがダウンした場合）。これは、GeoHA が設定されている場合にのみ発生します。リモートピアからのインデックスリカバリの図は、ローカルのレプリカセット全体がダウンした場合のリモートピアからの最初の同期を示しています。

ETCD

CDL は、DB サービス検出として *etcd*（オープンソースのキーバリューストア）を使用します。CDL ポッド（エンドポイント、スロット、またはインデックス）が開始、強制終了、またはシャットダウンされると、 状態のパブリッシュによってイベントが *etcd* に更新されます。

CDL エンドポイントは、スロットポッドまたはインデックスポッドがいつ起動または停止したかを知る必要があります。したがって、スロットポッドおよびインデックスポッドのイベントの通知に *etcd* で登録されます。

同様に、CDL スロットポッドまたはインデックスポッドは、そのピアレプリカポッド（同じ map-id を持つ）がいつ起動または停止したかを知る必要があります。したがって、対応するスロットまたはインデックス（同じ map-id を持つ）のポッドイベントの通知に *etcd* で登録されます。

これにより、通知はこれらのイベントに登録された各ポッドに送信されます。また、キーイベントが追加または削除されると、ローカルマップが更新されます。*etcd* キャッシュは、ローカルサイトのイベントにのみ適用されます。

Kafka

Kafka ポッドは、インデックス作成のためにローカルレプリカ間とサイト間でデータをレプリケートします。サイト間のレプリケーションの場合、Kafak は MirrorMaker を使用します。Kafka ポッドは、高可用性のためにレプリカ数が少なくとも 2 に設定されたセッション *vms* に展開されます。

Zookeeper

Kafka は Zookeeper ポッドを使用して Kafka クラスタを管理し、Kafka ブローカとの調整を行います。

Mirror Maker

Mirror Maker ポッドは、インデックスデータをリモート CDL サイトに Geo レプリケートします。リモートサイトからデータを取得し、ローカルの Kafka サイトにパブリッシュして、適切なインデックス作成インスタンスを取得します。

リモートサイトのモニタリング

機能説明

CDL エンドポイントは、CDL Ping RPC を使用して 30 秒ごとにリモートサイトの接続（レプリケーション接続と内部運用接続）をモニターします。いずれかの接続に対して ping が 3 回失敗した場合は、その接続を再作成して古い接続を閉じます。

リモートサイトのモニタリングは設定可能で、デフォルトでは有効になっています。**cdl datastore session features remote-site-connection-monitoring enable** CLI コマンドを使用して、リモートサイトの接続を有効または無効にします。

リモートサイト接続の設定

リモートサイトの接続を有効または無効にするには、次の CLI コマンドを使用します。

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

トラブルシューティング情報

エンドポイントのリモートサイト接続のログを表示するには、次の設定を使用します。

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

CDL 導入モデル

このセクションでは、次のようなさまざまな CDL 導入モデルについて説明します。

- CDL HA
- Geo HA デプロイメント

CDL HA デプロイメントには、次のものが含まれます。

1. 各 CDL ポッドは、高可用性を実現するために少なくとも 2 つのレプリカで構成されます。
2. CDL エンドポイントポッドは、サービス検出のために *etcd* を使用してローカルスロットとインデックスを検出します。
3. 3. インデックスおよびスロットポッドのデータは、書き込みの拡張性のためにマップ（またはシャード）をまたいで分散され、各マップは高可用性のために少なくとも 1 つのレプリカを備えます。スロットとインデックスのレプリカのデフォルトかつ推奨のレプリカ数は 2 です。
4. 要求を受信した CDL エンドポイントは、データを書き込むまたは読み取るスロットシャードを選択します。新しいレコードを作成する場合は、ラウンドロビン方式でスロットマップを選択します。既存のセッションの場合、スロットマップはインデックス作成でプライマリキーを検索した後に選択されます。
5. セッションで新しいキー（プライマリまたは一意）が追加または削除された場合、CDL エンドポイントはインデックス作成ポッドにキーを送信します。インデックスマップの選択は、キーをハッシュし、インデックス内のキーを更新するために適切なインデックスマッ

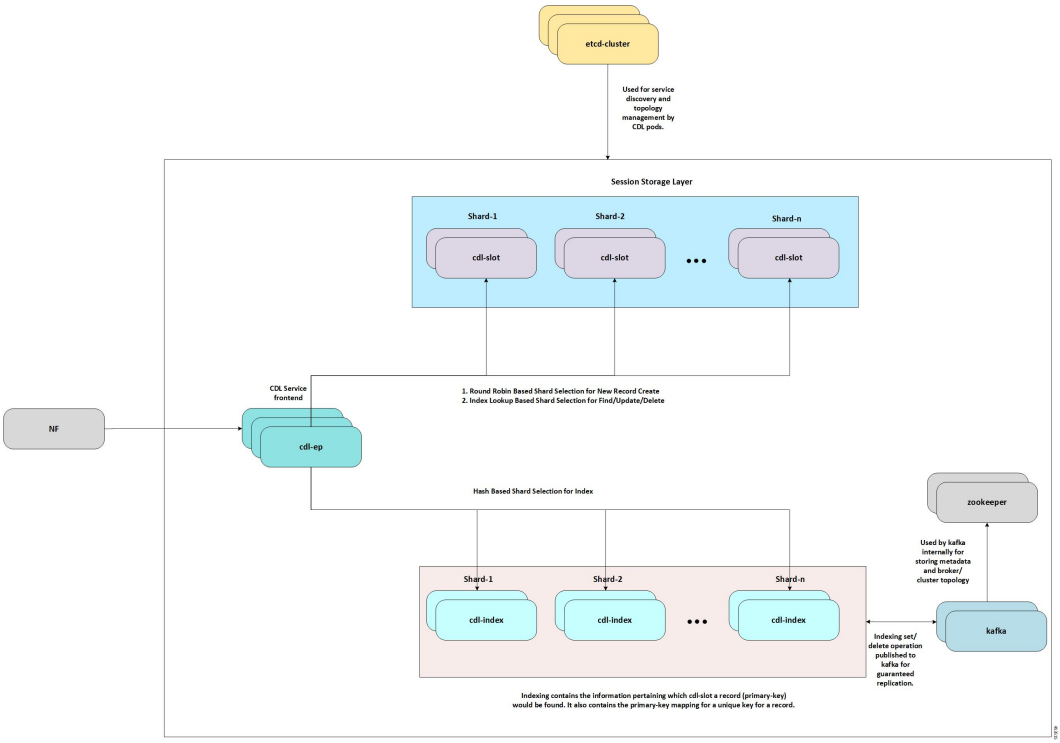
プに送信することによって行われます。プライマリキーの場合、インデックス作成ポッドはセッションが保存されているスロット map-id を保存します。一意のキーの場合、インデックス作成ポッドはセッションが保存されているプライマリキーを保存します。

6. CDLエンドポイントは、スロットメモリのデータを書き込むまたは読み取るために、選択されたスロットに作成、更新、削除、または検索要求を転送します。要求が正常に処理されると、スロットはエンドポイントに対して適切な応答を生成します。

7. 各インデックス作成シャードには、ローカルおよびリモートのレプリケーションを保証するために、インデックス作成操作を Kafka バスにパブリッシュするリーダーが含まれています。

8. 各インデックス作成ポッドインスタンスは、インデックス作成イベントのKafkaバスをリッスンします。設定または削除イベントを受信すると、イベントのタイムスタンプをすでに存在するインデックスのタイムスタンプと比較します。Kafka イベントのタイムスタンプが現在のインデックスのタイムスタンプよりも大きい場合、インデックス作成ポッドに操作が適用されます。それ以外の場合は、Kafka からのイベントは無視されます。

図 *9 : CDL HA* のデプロイメント



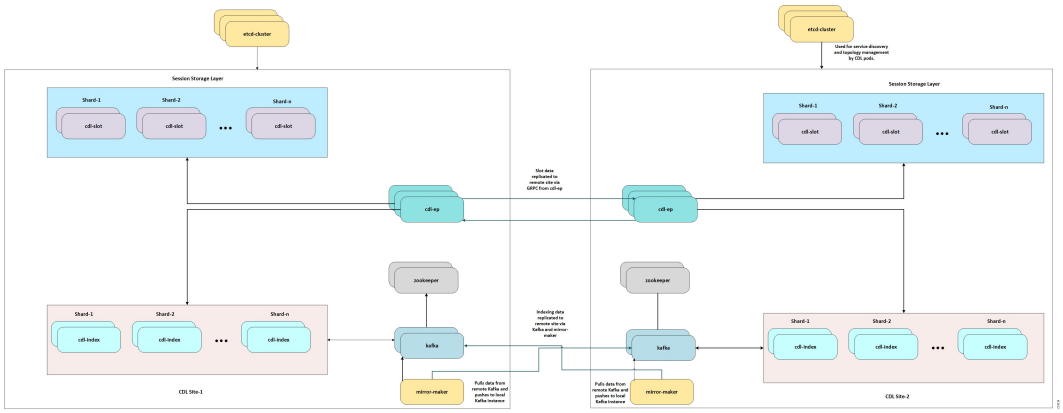
CDL Geo HA デプロイメントには、次のものが含まれます。

1. CDL Geo HA デプロイメントの場合、各サイトは Geo 冗長性を実現するためにリモートサイトで設定されます。

2. CDLエンドポイントは、ローカルのスロットとインデックスにデータを送信するだけでなく、GRPCを介してリモートサイトの *cdl-endpoint* にも要求を送信し、リモートサイトのスロットデータをレプリケートします。
3. リモートサイトで作成、更新、または削除要求を受信すると、CDLエンドポイントは、元のサイトで選択されていたものと同じ map-id で要求をスロットに転送します。
4. インデックス作成データのレプリケーションは、Mirror-maker によって実現されます。リモートサイトの Mirror-maker は、他のサイトの Kafka からのデータを消費し、そのデータをローカルのインデックスポッドにレプリケートするためにローカルの Kafka バスにプッシュします。

（注） Geo HA デプロイメントの場合、2 つの Geo サイト間の推奨ラウンドトリップ時間（RTT）は 50 ミリ秒です。

図 *10 : CDL Geo HA* のデプロイメント



コール フロー

このセクションでは、次のコールフローについて説明します。

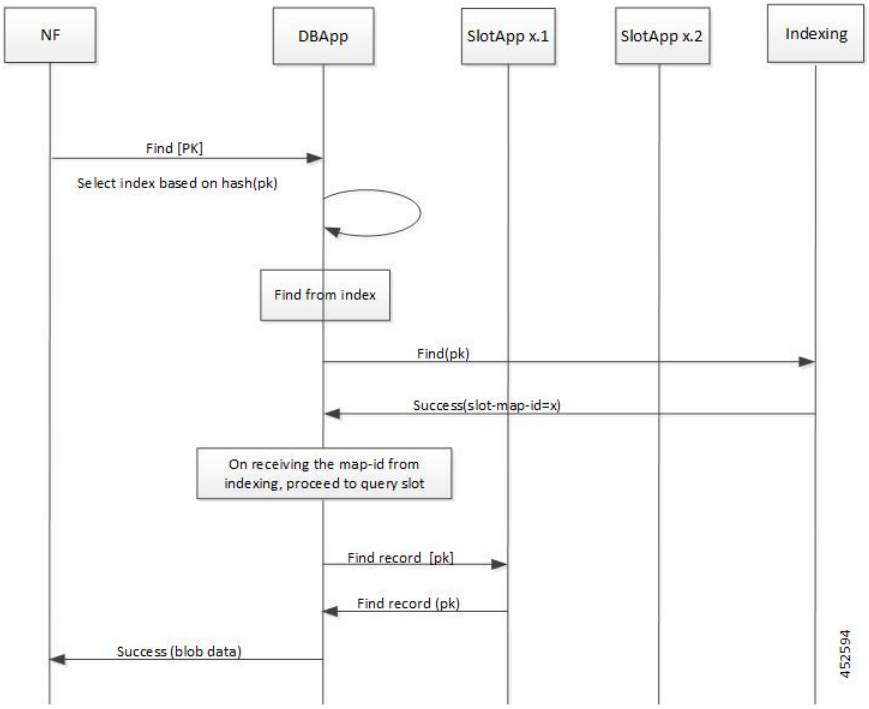
- プライマリキーによるレコードの検索
- 一意のキーによるレコードの検索
- レコードの作成
- レコードの更新
- レコードの削除
- タイマーが切れた時の NF への通知
- Geo レプリケーション：作成（GEO）

- Geo レプリケーション：更新（GEO）
- Geo レプリケーション：削除（GEO）

プライマリキーによるレコードの検索

このセクションでは、プライマリキーでレコードを検索するコールフローについて説明します。

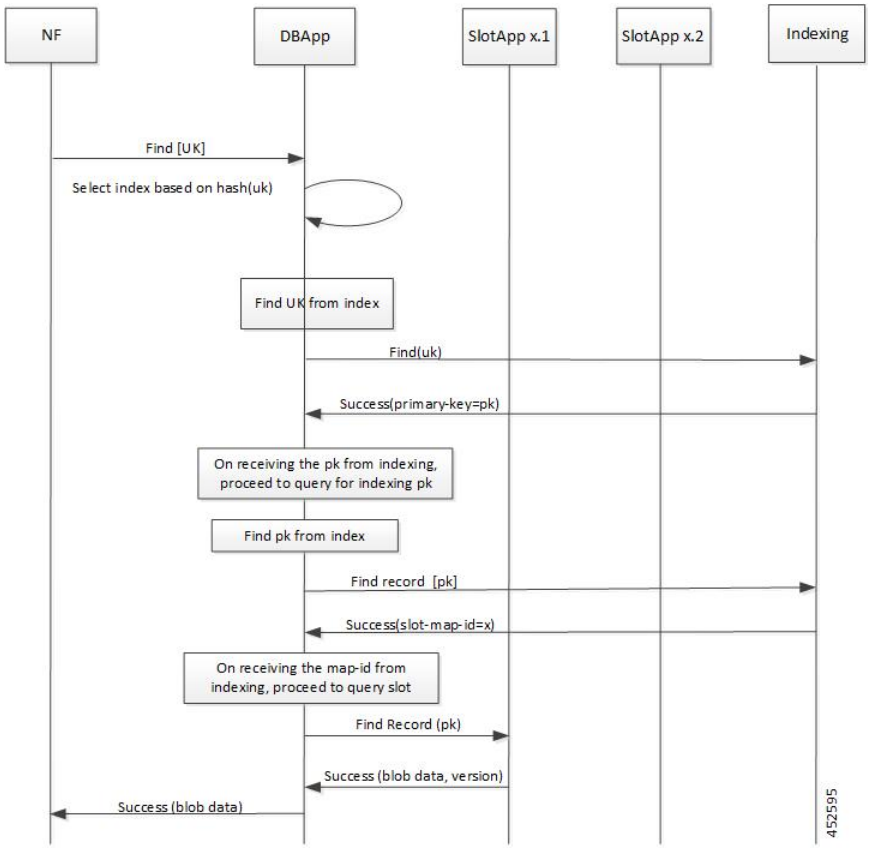
図 *11* : プライマリキーによるレコードの検索：コールフロー



一意のキーによるレコードの検索

このセクションでは、一意のキーでレコードを検索するコールフローについて説明します。

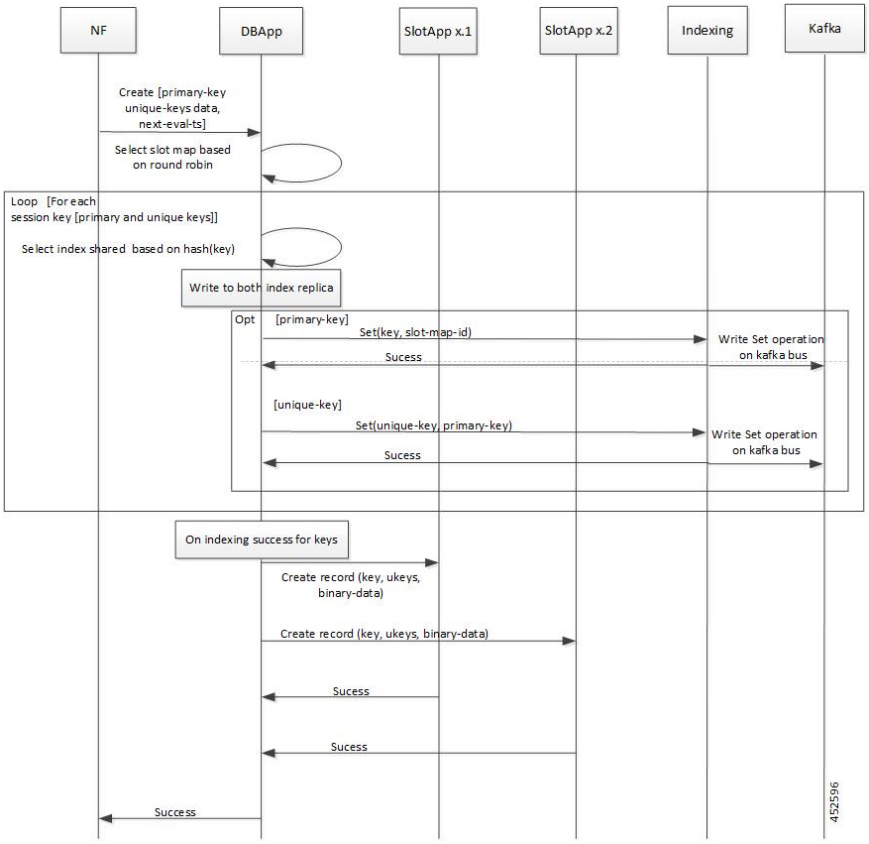
図 *12 :* 一意のキーによるレコードの検索：コールフロー



レコードの作成

このセクションでは、レコードを作成するコールフローについて説明します。

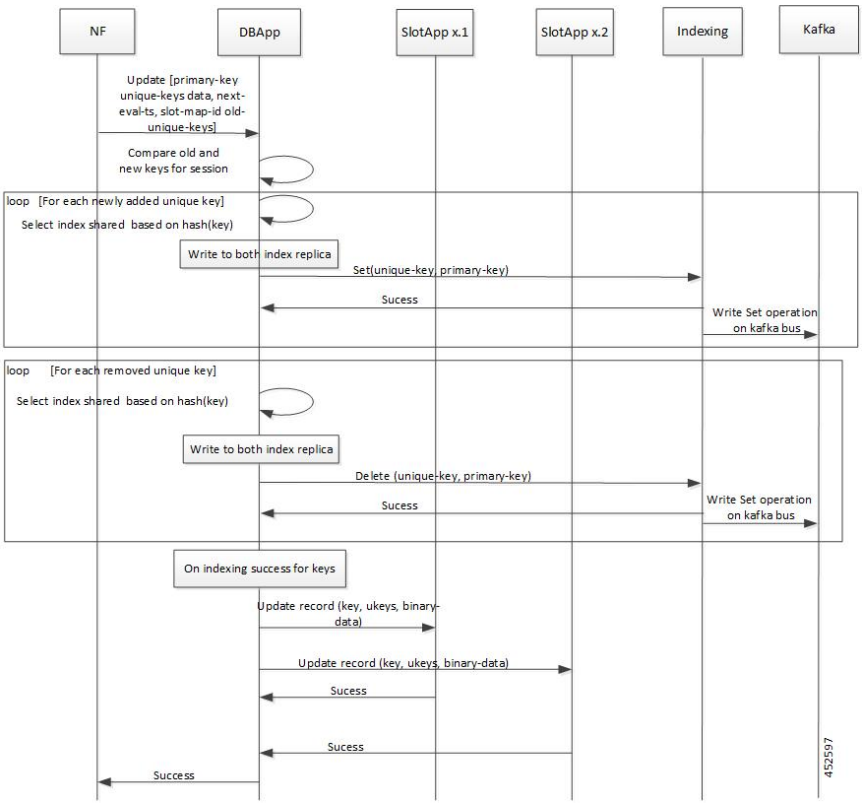
図 *13 :* レコードの作成：コールフロー



レコードの更新

このセクションでは、レコードの更新のコールフローについて説明します。

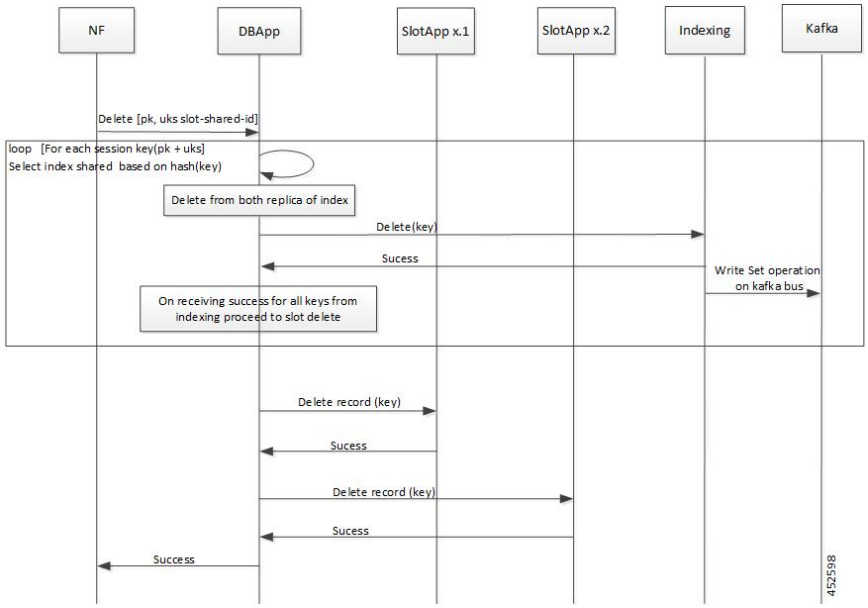
図 *14 :* レコードの更新：コールフロー



レコードの削除

このセクションでは、レコードの削除のコールフローについて説明します。

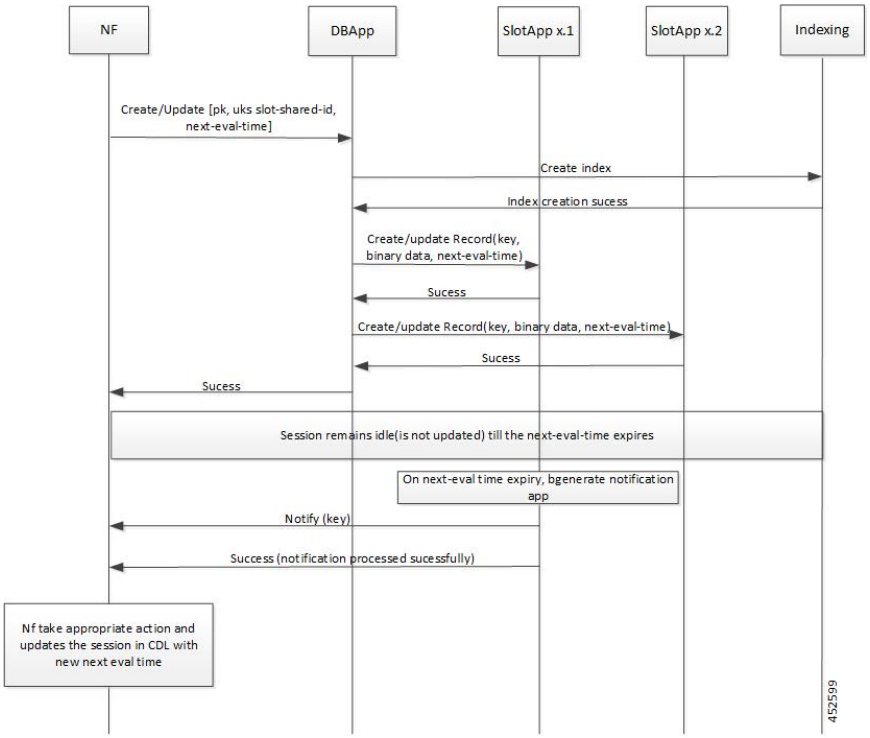
図 *15 :* レコードの削除：コールフロー



タイマーが切れた時の NF への通知

このセクションでは、タイマーが切れた時の NF へのセッション通知のコールフローについて説明します。

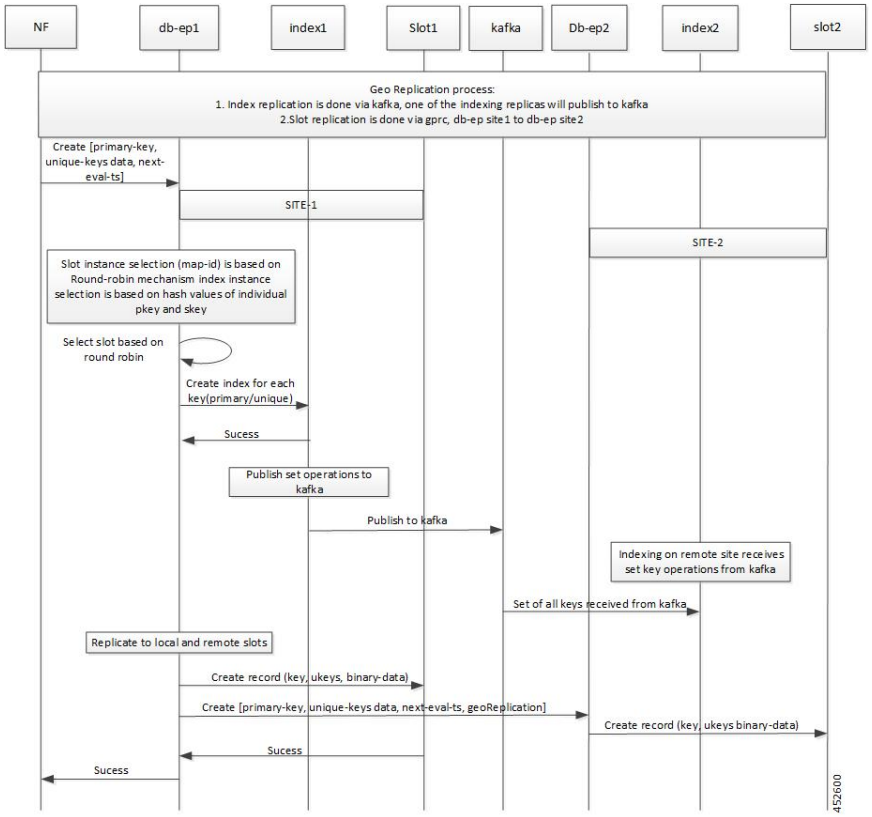
図 *16 : NF* へのセッション通知：コールフロー



Geo レプリケーション：作成

このセクションでは、Geo レプリケーションのプライマリキーと一意のキーを作成するコールフローについて説明します。

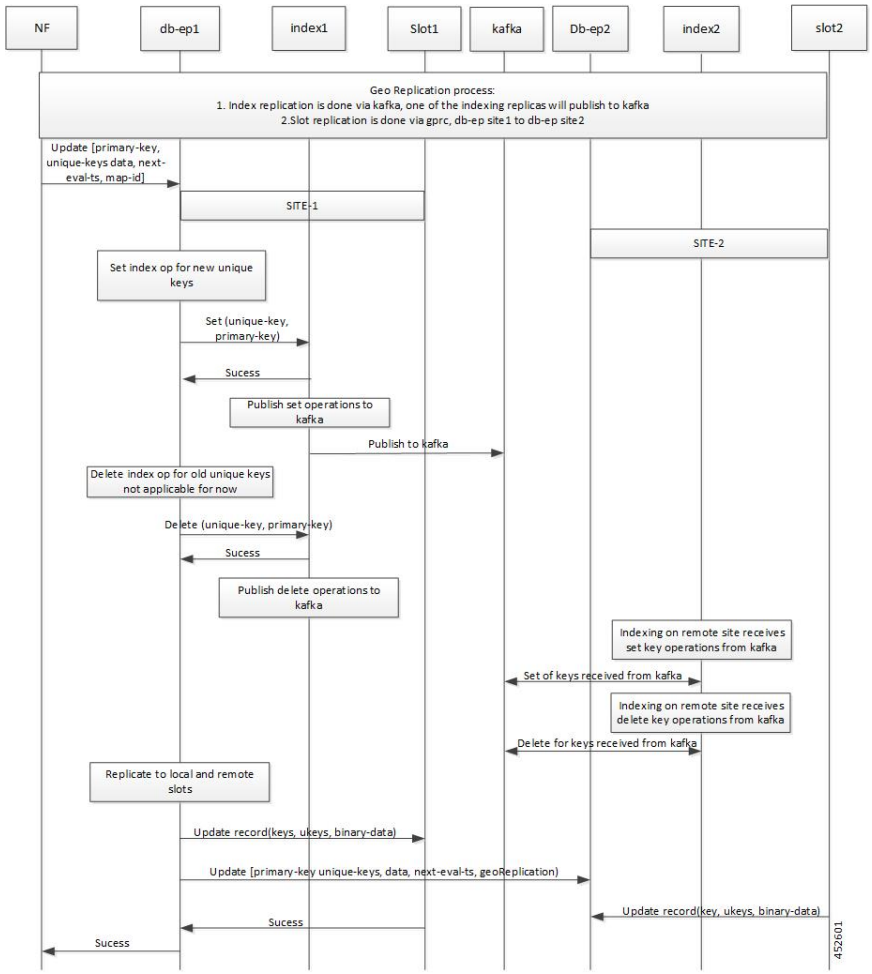
図 *17 : Geo* レプリケーションのコールフロー：*Geo* の作成



Geo レプリケーション：更新

このセクションでは、Geo レプリケーションのプライマリキーと一意のキーを更新するコールフローについて説明します。

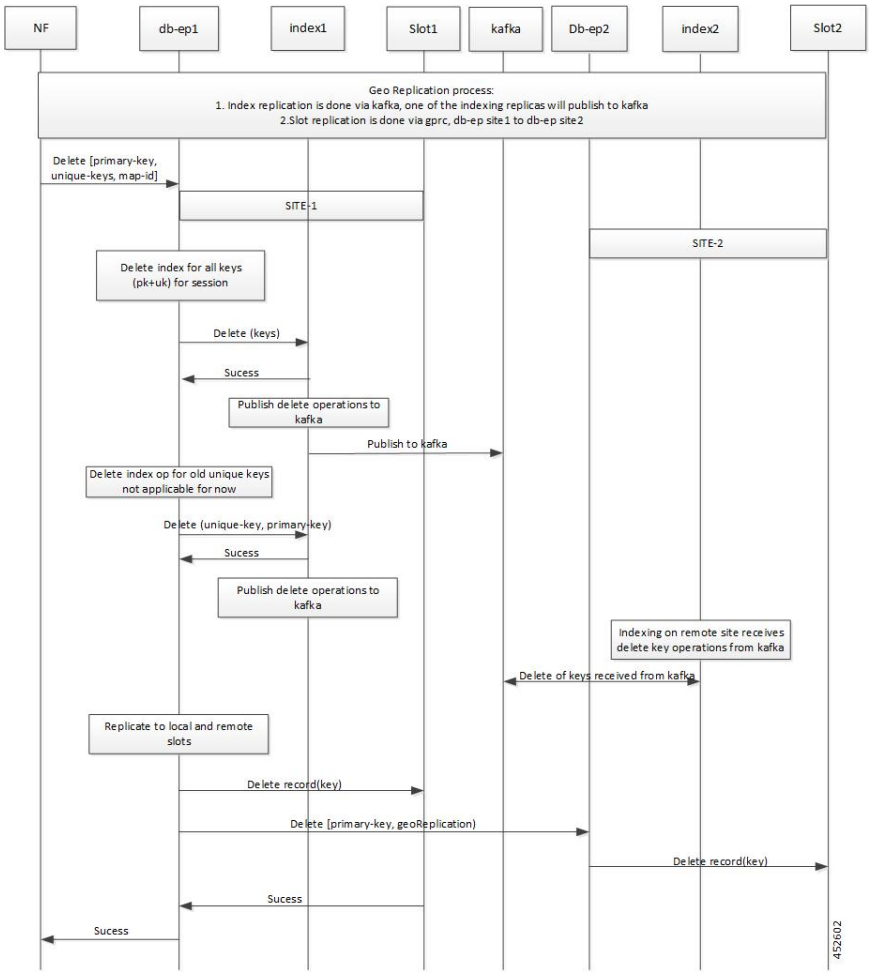
図 *18 : Geo* レプリケーションのコールフロー：*Geo* の更新



Geo レプリケーション：削除

このセクションでは、Geo レプリケーションのプライマリキーと一意のキーを削除するコールフローについて説明します。

図 *19 : Geo* レプリケーションのコールフロー：*Geo* の削除



古いインデックスレコードの識別

特定のシナリオでは、CDLのインデックスポッドの一意のキーが古くなっています（削除されると想定）。CDL は古いレコードの詳細を NF に示さないため、NF はこれらの一意のキーを別のレコードに使用しようとする場合があります。

CDL ではオペレータが、古いレコードを検出し、その古いレコードに対して必要なアクションを実行するパラメータを有効にできます。

古いレコードを検出してアクションを実行するには、次の手順を実行します。

1. 古いレコードを識別します。新しいレコードが作成されると、一意のキーは上書きされます。このようなレコードを識別するには、**index-overwrite-detection** パラメータを有効にし、*unique-keys-prefix* がプレフィックスパターンと一致することを確認します。
2. 識別された古いレコードに対して必要な**アクション**（通知、削除、またはログ）を実行します。

CDL は、上書きされた一意のキーを検出し、次のいずれかのアクションを実行します。

- 古いレコードまたはレコード全体を削除します。削除アクションは、古いレコードの **PurgeOnEval** フラグが false に設定されている場合にのみトリガーされます。
- 古いレコードについて NF に通知します。通知アクションは、**STALE_INDEX_NOTIFICATION** を NF に送信します。
- 上書きされた一意のキーをログに記録します。ログアクションの場合、古いレコードは WARN ログレベルでログに記録されます。

（注） 同じプライマリキーを指している2つの異なる一意のキーがあり、一方が通知アクションで、もう一方が削除アクションである場合、通知アクションが実行されます。

次の設定を使用します。

```
features index-overwrite-detection max-tps variable
features index-overwrite-detection queue-size variable
features index-overwrite-detection unique-keys-prefix uk
  action [delete-record, notify-record, log-record]
```

値は次のとおりです。

- **max-tps**：古いレコードの通知レートを制御します。デフォルトは 200 です。
- **queue-size**：古いレコードを処理するキューサイズを制御します。デフォルトは 1000 です。

（注） **queue-size** パラメータは、削除アクションと通知アクションの両方に使用されます。**max-tps** パラメータは通知アクション専用です。

- **unique-keys-prefix**：実行する必要があるアクションとともに固有のキーのプレフィックスパターンを指定します。

例：

```
cdl datastore session
features index-overwrite-detection max-tps 250
 features index-overwrite-detection queue-size 2000
 features index-overwrite-detection unique-keys-prefix uk
 action notify-record
 exit
  exit
```

トラブルシューティング

古いインデックスレコードをトラブルシュートするには、**index.overwrite.session** ロガーを INFO レベルに設定します。エンドポイントポッドおよびインデックスポッドからのログは、障害対応に役立ちます。

CDL の設定：

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

次のメトリックスが導入されています。

- **overwritten_index_records_deleted**：インデックスで特定された古いレコードが原因で削除されたレコードの総数を保持します。
- **overwritten_index_records_skipped**：古いレコードとして検出されたものの、そのレコードの通知または削除の処理中にキューがいっぱいになった場合はドロップされるレコードの総数を保持します。

ネットワーク機能（NF）を介した CDL の展開

ネットワーク機能（NF）（セッション管理機能（SMF）およびポリシー制御機能（PCF））の Ops Center を介して CDL を展開できます。

SMF Ops Center を介した CDL の展開については、『*Ultra Cloud Core 5G Session Management Function - Configuration and Administration Guide*』の「*Cisco Common Data Layer in SMF*」の章を参照してください。

PCF Ops Center を介した CDL の展開については、『*Ultra Cloud Core 5G Policy Control Function -Configuration and Administration Guide*』の「*Cisco Common Data Layer in PCF*」の章を参照してください。

CDL Geo レプリケーション（GR）の展開

このセクションでは、Geo レプリケーション（GR）の CDL を展開する方法について説明します。

CDL GR の前提条件

CDL GR を展開する前に、次の設定を行います。

- CDL セッションデータベースを設定し、基本設定を定義します。
- CDL 用の Kafka を設定します。
- CDL 用の Zookeeper を設定します。

CDL セッションデータベースの設定と基本設定の定義

このセクションでは、CDL セッションデータベースを設定し、NF（SMF または PCF）Ops Center を介して基本構成を定義する方法について説明します。

1. NF Ops Center コンソールを開き、データストア CLI に移動します。
2. セッションデータベースを設定し、CDL 操作のフェールオーバーの基本設定を定義するには、次の設定を使用します。

```
configure
   cdl system-id system_id
   cdl node-type node_type
   cdl enable-geo-replication boolean_value
   cdl remote-site remote_system_id db-endpoint host host_ip_address
      db-endpoint port port
      kafka-server remote_kafka_host remote_port
      end exit
cdl datastore session
      endpoint replica num_replica
      endpoint external-ip ip_address
      endpoint external-ip port
      slot replica num_replica
      slot map num_map/shards
      slot write-factor write_factor
      slot notification host host
      slot notification port port
      slot notification limit tps
      index replica num_replica
      index map num_map/shards
      index write-factor write_factor
      end exit
```

注：

- **cdl system-id** *system_id*：システムまたは Kubernetes クラスタの ID を指定します。デフォルト値は 1 です。
- **cdl node-type** *node_type*：ノードアフィニティを設定するための Kubernetes ノードラベルを示します。デフォルト値は session です。*node_type* は 0 ～ 64 文字の文字列である必要があります。
- **cdl enable-geo-replication** *boolean_value*：これはオプションの CLI です。Geo レプリケーションのステータスを有効または無効に指定します。デフォルト値は *false* です。
- **cdl remote-site** *remote_system_id*：リモート サイト エンドポイントのエンドポイントの IP アドレスを指定します。このコマンドは **cdl enable-geo-replication** を *true* に設定した場合にのみ設定します。
- **db-endpoint host** *host_ip_address*：リモートサイトのエンドポイント IP アドレスを指定します。このコマンドは **cdl enable-geo-replication** を *true* に設定した場合にのみ設定します。

- **db-endpoint port** *port_number*：リモート サイト エンドポイントのエンドポイントポートを示します。デフォルトのポート番号は 8882 です。このコマンドは **cdl enable-geo-replication** を *true* に設定した場合にのみ設定します。
- **kafka-server** *remote_kafka_host remote_port*：remote-system-id が識別する、リモートサイトの Kafka サーバーの外部 IP とポートを示します。リモートサイトの Kafka インスタンスごとに複数のホストとポートを設定できます。このコマンドは **cdl enable-geo-replication** を *true* に設定した場合にのみ設定します。
- **endpoint replica** *num_replica*：作成されるレプリカの数を示します。デフォルト値は 1 です。*num_replica* は 1 ～ 16 の範囲内にする必要があります。
- **endpoint external-ip** *ip_address*：これはオプションの CLI です。データベースエンドポイントを公開する外部 IP アドレスを指定します。このコマンドは **cdl enable-geo-replication** を *true* に設定した場合にのみ設定します。
- **endpoint external-port** *port*：データベースエンドポイントを公開する外部ポートを指定します。このコマンドは **cdl enable-geo-replication** を *true* に設定した場合にのみ設定します。デフォルト値は 8882 です。
- **slot replica** *num_replica*：作成するレプリカの数を指定します。デフォルト値は 1 です。*num_replica* は 1 ～ 16 の範囲内にする必要があります。
- **slot map** *num_map/shards*：スロット内のパーティション数を指定します。デフォルト値は 1 です。*num_map/shards* は 1 ～ 1024 の範囲内にする必要があります。
- **slot write-factor** *write_factor*：正常な応答の前に書き込まれるコピーの数を指定します。デフォルト値は 1 です。*write_factor* は 0 ～ 16 の範囲内にする必要があります。値がレプリカ数以下であることを確認してください。
- **slot notification host** *host*：通知サーバーのホスト名または IP アドレスを指定します。デフォルト値は *datastore-notification-ep* です。
- **slot notification port** *port*：通知サーバーのポート番号を指定します。デフォルト値は 8890 です。
- **slot notification limit** *tps*：1 秒あたりの通知上限を指定します。デフォルト値は 2000 です。
- **index replica** *num_replica*：作成するレプリカの数を指定します。デフォルト値は 2 です。*num_replica* は 1 ～ 16 の範囲内にする必要があります。
- **index map** *num_map/shards*：スロット内のパーティション数を指定します。デフォルト値は 1 です。指定できる範囲は 1 ～ 1024 です。CDL の展開後は、この値を変更しないでください。
- **index write-factor** *write_factor*：正常な応答の前に書き込まれるコピーの数を指定します。デフォルト値は 1 です。*write_factor* は 0 ～ 16 の範囲内にする必要があります。

CDL 用の Kafka の設定

このセクションでは、CDL 用の Kafaka の設定方法を説明します。

1. NF Ops Center のコンソールを開き、データストア CLI に移動します。
2. 次の設定を使用します。

```
configure
    cdl kafka replica num_replicas
        enable-JMX-metrics boolean_value
        external-ip ip_address port_number
        retention-time retention_period
        retention-size retention_size
        end
        exit
```

（注）

- **cdl kafka replica** *num_replicas*：作成するレプリカの数を指定します。デフォルト値は 3 です。*num_replicas* は 1 ～ 16 の範囲内である必要があります。
- **enable-JMX-metrics** *boolean_value*：JMX メトリックのステータスを指定します。デフォルト値は true です。
- **external-ip** *ip_address port_number*：Kafkaサービスに公開する外部 IP を指定します。**enable-geo-replication** パラメータを *true* に設定した場合に、このコマンドを設定します。Kafka レプリカの各インスタンスに外部 IP アドレスとポート番号を定義する必要があります。たとえば、**cdl kafka replica** パラメータが 3 に設定されている場合は、3 つの外部 IP アドレスとポート番号を定義する必要があります。
- **retention-time** *retention_period*：データを保持する必要がある期間（時間単位）を指定します。デフォルト値は 3 です。*retention_period* は 1 ～ 168 の範囲内である必要があります。
- **retention-size** *retention_size*：データの保持サイズを MB 単位で指定します。デフォルト値は 5120 MB です。

CDL 用の Zookeeper の設定

このセクションでは、CDL 用の Zookeeper の設定方法を説明します。

CDL 用の Zookeeper を設定するには、次の設定を使用します。

1. NF Ops Center のコンソールを開き、データストア CLI に移動します。
2. 次のコマンドを実行します

```
configure
 cdl zookeeper data_storage_size_in_gb
  log-storage-size size_in_gb
  replica num_replicas
  enable-JMX-metrics boolean_value
  enable-persistence boolean_value
  end
          exit
```

（注）

- **cdl zookeeper** *data_storage_size_in_gb*：Zookeeper データストレージのサイズを GB 単位で指定します。デフォルト値は 20 GB です。*data_storage_size_in_gb* は 1 ～ 64 の範囲内である必要があります。
- **log-storage-size** *size_in_gb*：Zookeeper データログのストレージのサイズを GB 単位で示します。デフォルト値は 20 GB です。*size_in_gb* は 1 ～ 64 の範囲内である必要があります。
- **replica** *num_replicas*：作成する必要があるレプリカの数を示します。デフォルト値は 3 です。*num_replicas* は 1 ～ 16 の範囲内である必要があります。
- **enable-JMX-metrics** *boolean_value*：JMX メトリックのステータスを示します。デフォルト値は true に設定されています。
- **enable-persistence** *boolean_value*：Zookeeper データの永続ストレージのステータスを指定します。デフォルト値は false です。

Geo レプリケーション（GR）用の CDL の展開

Geo レプリケーション（GR）では、2 つの独立した HA システムを作成し、リモートサイトとの通信用に Geo HA を設定する必要があります。デフォルトでは、CDL は *db-ep* に 2 つのレプリカ、1 つのスロットマップ（マップごとに 2 つのレプリカ）、および 1 つのインデックスマップ（マップごとに 2 つのレプリカ）で展開されます。

重要

- YANG モデルで CDL コンテナを設定することをお勧めします。
- Geo HA は、2 つの HA システムがアクティブになってから設定することをお勧めします。

CDL GR を展開するには、次の設定を使用します。

手順

ステップ 1 サイト 1 に HA を展開します。

Geo HA 関連の設定パラメータを使用せずにサイト 1 の HA システムを展開し、設定で **system-id** パラメータを *1* に設定します。

```
configure
    cdl system-id system_id
    cdl node-type node_label
    cdl datastore datastore_name
    endpoint replica number_of_HA_instances
    index map number_of_index_partitions
    index replica number_of_HA_instances
```

```
    slot map number_of_slot_partitions
    slot replica number_of_HA_instances
    exit
```

例：

```
cdl# configure terminal
cdl(config)# cdl system-id 2
cdl(config)# cdl node-type session
cdl(config)# cdl datastore session
cdl(config-datastore-session)# endpoint replica 2
cdl(config-datastore-session)# slot map 4
cdl(config-datastore-session)# slot replica 2
cdl(config-datastore-session)# index map 3
cdl(config-datastore-session)# index replica 2
cdl(config-datastore-session)# exit
cdl(config)#
```

ステップ 2 サイト 1 に HA 設定を適用します。

設定をコミットし、サイト 1 にポッドを展開します。

ステップ 3 サイト 2 に HA を展開します。

Geo HA 関連の設定パラメータを使用せずにサイト 2 の HA システムを（並行して）展開し、設定で **system-id** パラメータを *2* に設定します。

```
configure
    cdl system-id system_id
    cdl node-type node_label
    cdl datastore datastore_name
    endpoint replica number_of_HA_instances
    index map number_of_index_partitions
    index replica number_of_HA_instances
    slot map number_of_slot_partitions
    slot replica number_of_HA_instances
    exit
```

例：

```
cdl# configure terminal
cdl(config)# cdl system-id 2
cdl(config)# cdl node-type session
cdl(config)# cdl datastore session
cdl(config-datastore-session)# endpoint replica 2
cdl(config-datastore-session)# slot map 4
cdl(config-datastore-session)# slot replica 2
cdl(config-datastore-session)# index map 3
cdl(config-datastore-session)# index replica 2
cdl(config-datastore-session)# exit
cdl(config)#
```

ステップ 4 サイト 2 に HA 設定を適用します。

設定をコミットし、サイト 2 にポッドを展開します。

ステップ 5 サイト 1 とサイト 2 がアクティブであるかどうかを確認します。

Geo HA 設定に進む前に、CDL、Kafka、および Zookeeper ポッドがアクティブであるかどうかを確認します。

ステップ 6 サイト 2 の通信用にサイト 1 で外部 IP アドレスを設定します。

a) サイト 1 で Geo レプリケーションを有効にし、リモートサイトをサイト 1 の 2 として設定します。

```
configure
    cdl enable-geo-replication true/false
    cdl datastore session geo-remote-site list_of_geo_replication_sites
```

例：

```
cdl# configure terminal
cdl(config)# cdl enable-geo-replication true
cdl(config)# cdl datastore session geo-remote-site 2
```

b) CDL エンドポイントの外部 IP アドレスを、サイト 2 がアクセスできるように設定します。

```
configure
   cdl datastore session endpoint external-ip site_1_external_ip_address
```

c) すべての Kafka レプリカの外部 IP アドレスとポートを設定します。

たとえば、Kafka に 2 つのレプリカ（デフォルト）が設定されている場合、IP アドレスとポート番号の 2 つの異なるペアを指定します。

```
configure
   cdl kafka external-ip site_1_external_ip_address port1
   cdl kafka external-ip site_1_external_ip_address port2
```

d) （オプション）SSL/TLS 証明書を設定して、ローカルサイトの TLS サポートを有効にします。

TLS 証明書を設定すると、ローカルサイトでセキュアな接続と非セキュアな接続の両方を受け入れることができます。同様に、SSL 証明書を設定すると、ローカルサイトはリモートサイトとのセキュアな接続を確立できます。

（注）
証明書はすべて、複数行の生のテキスト形式です。証明書が無効な場合、サーバーは非セキュアな接続を続行します。

```
configure
  cdl ssl-config enable true
  cdl ssl-config ip site_1_external_ip_address
  cdl ssl-config certs site_1_external_ip_address ssl-key ssl_key
  cdl ssl-config certs site_1_external_ip_address ssl-crt ssl_crt
```

ステップ 7 サイト 1 の通信用にサイト 2 に外部 IP アドレスを設定します。

a) サイト 2 で Geo レプリケーションを有効にし、リモートサイトをサイト 2 の 1 として設定します。

```
configure
   cdl enable-geo-replication true/false
   cdl datastore session geo-remote-site list_of_geo_replication_sites
```

例：

```
cdl# configure terminal
cdl(config)# cdl enable-geo-replication true
cdl(config)# cdl datastore session geo-remote-site 1
```

b) CDL エンドポイントの外部 IP アドレスを、サイト 1 がアクセスできるように設定します。

```
configure
   cdl datastore session endpoint external-ip site_2_external_ip_address
```

c) すべての Kafka レプリカの外部 IP アドレスとポートを設定します。

たとえば、Kafkaに2つのレプリカ（デフォルト）が設定されている場合、IPアドレスとポート番号の2 つの異なるペアを指定します。

```
configure
    cdl kafka external-ip site_2_external_ip_address port1
    cdl kafka external-ip site_2_external_ip_address port2
```

d) （オプション）SSL/TLS 証明書を設定して、ローカルサイトの TLS サポートを有効にします。

TLS 証明書を設定すると、ローカルサイトでセキュアな接続と非セキュアな接続の両方を受け入れることができます。同様に、SSL 証明書を設定すると、ローカルサイトはリモートサイトとのセキュアな接続を確立できます。

（注）
証明書はすべて、複数行の生のテキスト形式です。証明書が無効な場合、サーバーは非セキュアな接続を続行します。

```
configure
  cdl ssl-config enable true
  cdl ssl-config ip site_1_external_ip_address
  cdl ssl-config certs site_1_external_ip_address ssl-key ssl_key
  cdl ssl-config certs site_1_external_ip_address ssl-crt ssl_crt
```

ステップ 8 各サイトにリモートサイト情報を追加します。

a) サイト 1 でリモートサイトの *cdl-ep*（CDL エンドポイント）の設定を行います。

```
configure
    cdl remote-site 2 db-endpoint host site_2_cdl_ep_ip
```

b) サイト 1 でリモートサイトの Kafka の設定を行います。

```
configure
    cdl remote-site 2 kafka-server site_2_kafka1_ip_address site_2_kafka1_port
    cdl remote-site 2 kafka-server site_2_kafka2_ip_address site_2_kafka2_port
```

c) （オプション）SSL 証明書を設定して、サイト 1 でリモートサイトとのセキュアな接続を確立します。

（注）
証明書はすべて、複数行の生のテキスト形式です。証明書が無効な場合、サーバーは非セキュアな接続を続行します。

```
configure
  cdl ssl-config certs site_2_external_ip_address ssl-key ssl_key
  cdl ssl-config certs site_2_external_ip_address ssl-crt ssl_crt
```

サイト 1 でその設定をコミットします。

d) サイト 2 でリモートサイトの *cdl-ep* の設定を行います。

```
configure
   remote-site 1 db-endpoint host site_1_cdl_ep_ip_address
```

e) サイト 2 でリモートサイトの Kafka の設定を行います。

```
configure
   cdl remote-site 1 kafka-server site_1_kafka1_ip_address site_1_kafka1_port
   cdl remote-site 1 kafka-server site_1_kafka2_ip_address site_1_kafka2_port
```

f) （オプション）SSL 証明書を設定して、サイト 2 でリモートサイトとのセキュアな接続を確立します。

（注）
証明書はすべて、複数行の生のテキスト形式です。証明書が無効な場合、サーバーは非セキュアな接続を続行します。

```
configure
  cdl ssl-config certs site_1_external_ip_address ssl-key ssl_key
  cdl ssl-config certs site_1_external_ip_address ssl-crt ssl_crt
```

サイト 2 でその設定をコミットします。

ステップ 9 Mirror Maker ポッドが展開されているかどうか、および他のすべてのポッドがアクティブであるかどうかを確認します。

注：

- **cdl kafka external-ip** ：すべての Kafka レプリカの外部 IP アドレスとポートを指定します。
  - *site_external_ip_address*：リモートサイトの外部 IP アドレスを指定します。
  - *port_number*：リモートサイトのポート番号を指定します。
- **cdl ssl-config certs**：SSL 証明書を指定して、リモートサイトとのセキュアな接続を確立します。
  - *site_external_ip_address*：リモートサイトの外部 IP アドレスを指定します。
  - **ssl-key***ssl_key*：生成された SSL キーを指定します。

Geo レプリケーション（GR）のフェールオーバーの通知

CDL には Geo レプリケーション（GR）フェールオーバー通知が用意されていているため、セッションデータのタイマー切れの通知や、現在アクティブなサイトへの一括通知が可能です。CDL は GR フェールオーバー通知のために、App-Infra を介してボーダー ゲートウェイ プロトコル（BGP）を使用します。

CDL は両方の GR サイトでキー値を登録します。登録したキー値に変更があると、App-Infra は CDL に通知を送信します。キー値は、CDL システム ID または GR インスタンスの状態を示

します。GR インスタンスは、 キーの CDL システム ID または GR インスタンス ID を使用して CDL スライスにマップされます。

CDL GR フェールオーバー通知に、次のパラメータが導入されました。

- **instance-aware-notification enable *true or false***：GR フェールオーバー通知を有効にするには、これを true に設定します。
- **instance-aware-notification system-id** ：system-id は sliceName にマップされます。つまり、プライマリ system-id はその特定の sliceName のプライマリサイト ID にマップされます。この情報は、すべての Geo サイトで構成する必要があります。*FindAllNotify* 通知および *TimerExpiry* 通知は、一括処理タスクのシステム ID の詳細を使用します。

システム ID は両方のサイトで必須です。NF 構成の GR インスタンス ID が CDL システム ID と一致している必要があります。

次の例では、GR フェールオーバー通知が true に設定されています。system-id 1 は slicesName *sgw1* および *smf1* のプライマリサイト ID で、system-id 2 は slicesName *sgw2* および *smf1* のプライマリサイトです。

```
cdl datastore session
 features instance-aware-notification enable true
 features instance-aware-notification system-id 1
  slice-names [ sgw1 smf1 ]
 exit
 features instance-aware-notification system-id 2
  slice-names [ sgw2 smf2 ]
 exit
exit
```

ピアサイトの GR フェールオーバー通知レコード

CDL は、ローカルアプリケーションにのみレコードに関する通知を送信します。通知は、レコードの system-id パラメータと Timer Expiry パラメータに基づいています。GR セットアップでは、メンテナンスのためにサイトが分離されると、そのサイトのレコードはピアサイトに送信されません。

remote-system-id パラメータを使用すると、CDL はピアサイトが分離されたサイトの通知を処理することを許可します。ピアサイトの **remote-system-id** は、分離されたサイトのサイト ID で設定されます。レコードの system-id が remote-system-id と一致する場合、CDL はレコードを処理します。通知は、Timer Expiry に基づいて、または **notifyOnPurge** が有効になっているレコードに関して送信されます。

（注） 分離されたサイトの機能が復帰した後は、その remote-system-id を CDL 設定から削除する必要があります。

次の手順では、例を示してこの remote-system-id の設定について説明します。

次の例では、GR セットアップに 2 つのサイト（サイト 1 とサイト 2）があります。サイト 1 はアップグレードのために切断され、サイト 2 の remote-system-id がサイト 1 のサイト ID で設定されます。

1. サイト 1 をシャットダウンまたは切断します。
2. サイト 1 のサイト ID を使用してサイト 2 の **remote-system-id** を設定するには、次のコマンドを実行します。

```
cdl datastore session
    slot notification remote-system-id [ 1 ]
exit
```

上記のコマンドの **remote-system-id** の値 [1] は、分離されているサイト 1 のサイト ID であることに注意してください。

3. サイト 2 は、ローカルアプリケーションへのサイト 1 のレコードの通知を開始します。
4. サイト 1 を起動する前に、サイト 2 の remote-system-id リストからサイト 1 のサイト ID を削除します。

remote-system-id は、**instance-aware-notification-system-id** と相互に排他的です。詳細については、「*Geo* レプリケーション（*GR*）のフェールオーバーの通知」 のトピックを参照してください。

リモートインデックス同期のトリガー

CDLは、インデックスをリモートピアと同期させるためのユーティリティを提供します。このユーティリティは、サイト間でインデックスレコードの数に大きな違いがある場合の、サイト分離後などのシナリオで使用できます。CDLは、リモートインデックス同期のステータスを確認するコマンドもサポートしています。

（注）
- パフォーマンスに影響を与える可能性があるため、リモート同期はサイト間のインデックスレコードに大きな違いがある場合にのみ使用することをお勧めします。
- コマンドを表示するには、geo-remote-site を設定する必要があります。

以下に関するリモートインデックスの同期をトリガーします。

- インデックス mapID のリスト、または CLI からのすべての mapID。
- sliceName のリスト、またはすべての sliceName。

次の点に注意してください。

- インデックスのピアとのリモート同期のトリガー中のエラーを回避するために、次の条件を満たしていることを確認してください。
  - カスタムの map-id と slice-name は一意にする必要があります。
  - 有効なインデックス map-id と slice-name のみが許可されます。
  - リモートサイトは到達可能です。

- 内部的に、CDL はリモートピアとの同期のために最大 3 回再試行します。
- インデックスインスタンスの同期がすでに進行中の場合、リモートインデックス同期はそのインスタンスをスキップします。

コマンド	出力パラメータ
cdl actions remote-index-sync start *[options]* [オプション（Options）] • **map-id** リモートインデックス同期を開始する必要があるインデックス map-id。map-id はオプションです。含めると、そのインデックス map-id のすべてのインスタンスのリモートインデックス同期がトリガーされます。このオプションを使用して、最大 5 つの map-id を指定できます。 • **slice-name** リモートインデックス同期を開始する slice-name。slice-name はオプションです。含めると、すべての slice-name のリモートインデックス同期がトリガーされます。slice-name の数に制限はありません。	**triggered-instances** リモートインデックス同期が開始されているインデックスインスタンスのリストを表示します。
例	出力
cdl actions remote-index-sync start map-id { 1 } map-id { 2 } slice-name { session-1 } slice-name { session-2 }	**triggered-instances 'index-mapID-1-instanceID-1, index-mapID-1-instanceID-2, index-mapID-2-instanceID-1, index-m**

リモート同期ステータスの確認

リモート同期ステータスには、リモートインデックス同期が進行中のインデックスインスタンスが表示されます。

（注） リモート同期ステータスには、sliceName ごとの同期ステータスは表示されません。同期ステータスに特定のインデックスインスタンスがリモートサイトと同期していることが示されている場合は、すべての sliceName またはリストされている sliceName が順番に同期していることを意味します。

コマンド	出力パラメータ
cdl actions remote-index-sync status	**syncing-instances** リモートインデックス同期が進行中のインデックスインスタンスのリスト。
例 **cdl actions remote-index-sync status**	出力 **syncing-instances 'index-mapID-1-instanceID-1, index-mapID-1-instanceID-2, index-mapID-2-instanceID-1, index-mapID-2-instanceID-2'**

トラブルシューティング

インデックスポッドの次の警告またはエラーログには、リモートインデックス同期のステータスが示されます。

- 同期が成功している場合は、次のログが表示されます。

  ログ：

  Bulk Sync done from Remote Indexes(s) for isInitSync = false

  例：

  ```
  [datastore.index.session] Bulk Sync done from Remote Indexes(s) for isInitSync =
  false sliceName = session via DBApp
  ```

  ログ：

  Sync finished successfully with Remote MemoryMap App isInitSync: false

  例：

  ```
  [datastore.index.session] Sync finished successfully with Remote MemoryMap App
  isInitSync: false count: 100 Time taken: 16.699031ms sliceName: session
  ```

- 同期が失敗している場合は、次のエラーログが表示されます。

  例：

  ```
  Error! Remote Bulk Read failed 3 times for isInitSync = false
  ```

機能の概要と変更履歴

要約データ

該当製品または機能エリア	PCF 2021.04.0 以降
該当するプラットフォーム	ベアメタル、OpenStack、VMware
機能のデフォルト設定	無効：設定が必要

このリリースでの関連する変更点	N/A
関連資料	該当なし

マニュアルの変更履歴

改訂の詳細	リリース
最初の導入。	CDL 1.6

機能説明

このリリースでは、CDLはIPv6をサポートしているため、GRが有効になっているセットアップではエンドポイントのデュアルスタックサポートが有効になります。デュアルスタックにより、IPv4 アドレスと IPv6 アドレスの両方を使用してネットワーキングデバイスを設定できます。

この機能により、次のような機能が提供されます。

- CDL エンドポイントの IPv6 サポート。
- Mirror Maker と Kafka ブローカ間の通信用の IPv6 サポート。
- CDL および Kafka IPv6 エンドポイントの TLS サポート。

（注） CDL で IPv6 サポートを行うには、K8s クラスタがデュアルスタックをサポートしており、それがクラスタで有効になっている必要があります。

IPv6 サポートの設定

GR が有効になっているセットアップで CDL の IPv6 サポートを設定するには、次の手順で CLI コマンドまたは設定を使用します。

1. CDL IPv6 エンドポイントを公開し、リモートサイトの IPv6 エンドポイントに接続します。
   1. CDL Geo レプリケーションを有効にします。

      ```
      config
        cdl enable-geo-replication true
      ```

   2. CDL エンドポイントの外部 IP アドレスを設定します。**endpoint external-ip** に加えて、**endpoint external-ipv6** パラメータも設定します。

      ```
      config
        cdl datastore session endpoint external-ipv6 IPv6_address
      ```

   3. リモートサイトの CDL エンドポイントを設定します。

      ```
      config
        cdl remote-site system_id
      ```

```
    db-endpoint host IPv6_address
    db-endpoint port 8882
  exit
```

（注） IPv4 エンドポイントと IPv6 エンドポイントには、同じポート（デフォルト値は 8882）を介してアクセスできます。

（注） 現在、CDL は IPv4 または IPv6 のいずれかを使用してリモートサイトのエンドポイントに接続できますが、両方を使用することはできません。ただし、リモートサイトの IPv4 エンドポイントと IPv6 エンドポイントの両方を同時に公開して、いずれかに接続することはできます。

2. CDL Kafka を公開し、IPv6 を介してリモートサイトの Kafka に接続します。

   1. すべての Kafka レプリカの外部 IP アドレスとポートを設定します。

      ```
      config
        cdl kafka external-ipv6 IPv6_address port_address
        exit
      ```

（注） 各ブローカに個別に IPv6 を設定する必要があります。IPv4 と IPv6 を介して Kafka ブローカを同時に公開することは可能ですが、それぞれ異なるポートを使用する必要があります。Kafka リスナーは現在、異なる IP アドレスに対する同じポートの使用をサポートしていません。

   2. リモートサイトの Kafka 設定を行います。

      Mirror Maker は、IPv4 と IPv6 の両方またはいずれか 1 つを介して、ローカルサイトとリモートサイトの両方の Kafka ブローカに接続できます。

      次の CLI コマンドを使用して、リモートの Kafka サーバーを個別に設定します。

      ```
      config
        cdl remote-site system_id
          kafka-server IPv6_address port_address
        exit
      ```

3. （オプションの手順）SSL/TLS 証明書を設定して、ローカルサイトとリモートサイトの両方で TLS サポートを有効にします。これらの証明書は、ローカルサイトとリモートサイト間のセキュアな接続を確立するのに役立ちます。

   同じ設定を使用して、各サイトの証明書を設定します。

   1. CDL エンドポイントの SSL を有効にします。

      ```
      config
        cdl ssl-config enable true
      ```

2. 証明書を設定します。

```
config
  cdl ssl-config certs external_ipv6_address
    ssl-key ssl_key
    ssl-crt ssl_crt
    ssl-ca ssl_ca
  exit
```

注：

- **ssl-key** *ssl_key*：サーバーの秘密キーを指定します。
- **ssl-crt** *ssl_crt*：CA 署名付きサーバー証明書を指定します。
- **ssl-ca** *ssl_ca*：パブリック CA 証明書を指定します。

4. （オプションの手順）SSL/TLS 証明書を設定して、Kafka エンドポイントの TLS サポートを有効にします。

1. CDL エンドポイントの SSL を有効にします。

```
config
  cdl ssl-config enable true
```

2. Kafka エンドポイントの SSL を有効にします。

```
config
  cdl kafka ssl-settings enable-ssl true
  cdl kafka ssl-settings disable-host-name-verification true
```

3. ローカルサイトとリモートサイトの両方で Kafka エンドポイントの証明書を設定します。

```
config
  cdl ssl-config certs external_ipv6_address
    ssl-key ssl_key
    ssl-crt ssl_crt
    ssl-ca ssl_ca
  exit
```

次の設定例は、リモートサイトの IPv6 TLS エンドポイントと IPv6/IPv4 TLS Kafka に接続するために使用します。

サイト 1 の設定

```
cdl remote-site 2
 db-endpoint host 2001:DB8:54ff:a4::139:250
 db-endpoint port 8882
 kafka-server 10.106.139:250 10092
 ssl-port 10094
 exit
 kafka-server 10.106.139:250 10093
 ssl-port 10095
 exit
 kafka-server 2001:DB8:54ff:a4::139:250 10096
 ssl-port 10098
```

```
 exit
 kafka-server 2001:DB8:54ff:a4::139:250 10097
 ssl-port 10099
 exit
exit
cdl ssl-config enable true
cdl ssl-config certs 2001:DB8:54ff:a4::139:250
    ssl-key  "<server-key>"
    ssl-crt  "<signed-certificate>"
    ssl-ca  "<ca-certificate>"
exit
cdl ssl-config certs 192.0.2.2
    ssl-key  "<server-key>"
    ssl-crt  "<signed-certificate>"
    ssl-ca  "<ca-certificate>"
exit
cdl ssl-config certs 192.0.2.1
    ssl-key  "<server-key>"
    ssl-crt  "<signed-certificate>"
    ssl-ca  "<ca-certificate>"
exit
cdl ssl-config certs 2001:DB8:54ff:a4::139:249
    ssl-key  "<server-key>"
    ssl-crt  "<signed-certificate>"
    ssl-ca  "<ca-certificate>"
exit
cdl datastore session
 geo-remote-site [ 2 ]
 endpoint external-ip 192.0.2.1 endpoint external-ipv6 2001:DB8:54ff:a4::139:249
exit
cdl kafka ssl-settings enable-ssl true
cdl kafka ssl-settings disable-host-name-verification true
cdl kafka external-ipv6 2001:DB8:54ff:a4::139:249 10096
 ssl-port 10098
exit
cdl kafka external-ipv6 2001:DB8:54ff:a4::139:249 10097
 ssl-port 10099
exit
cdl kafka external-ip 10.106.139.249 10092
 ssl-port 10094
exit
cdl kafka external-ip 10.106.139.249 10093
 ssl-port 10095
exit
```

サイト 2 の設定

```
cdl remote-site 1
 db-endpoint host 2001:DB8:54ff:a4::139:249
 db-endpoint port 8882
 kafka-server 10.106.139:249 10092
 ssl-port 10094
 exit
 kafka-server 10.106.139:249 10093
 ssl-port 10095
 exit
 kafka-server 2001:DB8:54ff:a4::139:249 10096
 ssl-port 10098
 exit
 kafka-server 2001:DB8:54ff:a4::139:249 10097
 ssl-port 10099
 exit
exit
cdl ssl-config enable true
cdl ssl-config certs 2001:DB8:54ff:a4::139:249
```

```
    ssl-key  "<server-key>"
    ssl-crt  "<signed-certificate>"
    ssl-ca  "<ca-certificate>"
exit
cdl ssl-config certs 192.0.2.1
    ssl-key  "<server-key>"
    ssl-crt  "<signed-certificate>"
    ssl-ca  "<ca-certificate>"
exit
cdl ssl-config certs 192.0.2.1
    ssl-key  "<server-key>"
    ssl-crt  "<signed-certificate>"
    ssl-ca  "<ca-certificate>"
exit
cdl ssl-config certs 2001:DB8:54ff:a4::139:250
    ssl-key  "<server-key>"
    ssl-crt  "<signed-certificate>"
    ssl-ca  "<ca-certificate>"
exit
cdl datastore session
 geo-remote-site [ 1 ]
 endpoint external-ip 192.0.2.1
 endpoint external-ipv6 2001:DB8:54ff:a4::139:250
exit
cdl kafka ssl-settings enable-ssl true
cdl kafka ssl-settings disable-host-name-verification true
cdl kafka external-ipv6 2001:DB8:54ff:a4::139:250 10096
 ssl-port 10098
exit
cdl kafka external-ipv6 2001:DB8:54ff:a4::139:250 10097
 ssl-port 10099
exit
cdl kafka external-ip 192.0.2.1 10092
 ssl-port 10094
exit
cdl kafka external-ip 192.0.2.1 10093
 ssl-port 10095
exit
```

CDL アラーム

このセクションでは、CDL デプロイメントのアラームについて説明します。

障害を検出するために次のアラームが導入されています。

表 *2:* アラーム

アラーム	シビラティ（重大度）	説明
cdlLocalRequestFailure	critical	このアラームは、ローカル要求の成功率が 5 分以上にわたって 90% 未満の場合にトリガーされます。

アラーム	シビラティ（重大度）	説明
cdlRemoteConnectionFailure	critical	このアラームは、エンドポイントポッドからリモートサイトへのアクティブな接続が 5 分以上にわたって 0 に達した場合にトリガーされます。このアラームは GR デプロイメント専用です。
cdlRemoteRequestFailure	critical	このアラームは、受信リモート要求の成功率が 5 分以上にわたって 90% 未満の場合にトリガーされます。このアラームは GR デプロイメント専用です。
cdlReplicationError	critical	cdl-global 名前空間でのローカル要求に対する発信レプリケーション要求の比率が 5 分以上にわたって 90% 未満の場合。このアラームは GR デプロイメント専用です。
cdlKafkaRemoteReplicationDelay	critical	このアラームは、リモートサイトへの Kafka レプリケーションの遅延が 5 分以上にわたって 10 秒を超えた場合にトリガーされます。このアラームは GR デプロイメント専用です。

CDL ゾーンのアップグレード

機能説明

CNDP は、アップグレードゾーン内のすべてのノードが並行してアップグレードされる、ゾーンベースのアップグレードをサポートしています。これにより、大規模なクラスタのアップグレードに時間をかけずに済みます。CDL は、K8s のトポロジ分散制約機能を利用して、アップグレードゾーン全体にポッドを分散します。

CDL でこのインサービスをサポートするには、ポッドがアップグレードゾーンを認識している必要があり、スロット、インデックス、または Kafka のすべてのレプリカが同じアップグレードゾーンでスケジュールされないようにする必要があります。たとえば、同じマップの両方のスロットレプリカが同じアップグレードゾーンにある場合、両方のレプリカが同時にダウンし、非 GR シナリオでセッションの損失を引き起こします。CDL エンドポイントは、レプリカが 1 つ戻るまで要求の処理を停止します。

少なくとも 3 つのアップグレードゾーンが存在することを確認してください。ノード障害が発生すると、別のゾーンにポッドをスケジュールするスペースがある場合でも、ポッドが保留状態になる可能性があります。

この機能を有効にするために CDL または NF Ops Center で必要な明示的な設定はありません。クラスタデプロイヤ自体が設定を渡します。

トポロジの分散制約

CDL は、K8s トポロジの分散制約機能を使用してアップグレードゾーン全体にポッドを分散し、アップグレードゾーンがダウンした場合にすべての冗長ポッドが同時に失われないようにします。

トポロジの分散制約について、次のフィールドが定義されています。

- **maxSkew**：ターゲットトポロジで一致するポッドの数とグローバル最小数（トポロジドメインのラベルセレクタに一致するポッドの最小数）との間で許容される最大差を定義します。たとえば、それぞれ 2 つ、4 つ、5 つの一致するポッドを持つ 3 つのゾーンがある場合、グローバル最小数は 2 となり、maxSkew はその数に対して比較されます。
- **whenUnsatisfiable**：分散制約を満たさない場合のポッドの処理方法を示します。
  - **DoNotSchedule**（デフォルト）は、スケジューラがポッドをスケジュールしないことを示します。
  - **ScheduleAnyway** は、スケジューラがスキューを最小限に抑えるノードに優先順位を付けながらポッドをスケジュールすることを示します。
- **topologyKey** は、ノードラベルのキーです。2 つのノードがこのキーでラベル付けされ、そのラベルの値が同じである場合、スケジューラは両方のノードを同じトポロジ内にあるものとして扱います。スケジューラは、バランスの取れた数のポッドを各トポロジドメインに配置しようとします。
- **labelSelector** を使用して、一致するポッドを検索します。このラベルセレクタに一致するポッドをカウントし、対応するトポロジドメイン内のポッドの数を決定します。

CDL ポッドについて、トポロジ分散制約のために定義されたフィールドの例を次に示します。

- **topologyKey: smi.cisco.com/upgrade-zone**
- **maxSkew: 1**
- **whenUnsatisfiable: DoNotSchedule**
- **labelSelector:** デプロイメントまたはステートフルセットに一致するすべてのポッドを選択します。

K8s スケジューラはこの制約に基づき、maxSkew が 1 のゾーン全体へのデプロイメントまたはステートフルセットでポッドを均等に分散しようとします。

- 2 つのレプリカを持つスロットまたはインデックスポッドでは、2 つのレプリカは常に別々のアップグレードゾーンにあります。
- cdl-ep または Kafka ポッドの場合、ポッドは maxSkew が 1 の状態でほぼ均等に分散されます。

- ポッドが **maxSkew: 1** 制約を満たすことができない場合、**whenUnsatisfiable** フィールドが **DoNotSchedule** に設定されるため、そのポッドは保留状態のままになります。この制約は、maxSkew の違反を許可しません。

（注） トポロジの分散制約は、cdl-ep、cdl-slot、インデックス、Kafka、またはステートフルセットのデプロイメントにのみ追加されます。etcd と zookeeper は、1 つずつアップグレードされるプライマリノードまたは OAM ノードでスケジュール設定されるため、変更はありません。

制限事項と制約事項

このセクションでは、CDL ゾーンアップグレードの既知の制限事項と制約事項について説明します。

- CDL は、Kafka レプリカ 2 または Kafka レプリカ 3 を使用した NF のデプロイメントをサポートしています。Kafka レプリカ 3 を使用したデプロイメントの場合、CDL は少なくとも 3 つのアップグレードゾーンを定義します。3 つのレプリカと 2 つのアップグレードゾーンがある場合、一方のアップグレードゾーンに 2 つのレプリカがあり、もう一方のアップグレードゾーンに 1 つのレプリカがあります。2 つのレプリカを持つアップグレードゾーンがダウンしたこのシナリオでは、1 つのレプリカのみがトラフィックを処理できます。
- アップグレードゾーン内のノードがすべて使用されている場合、ゾーン内の 1 つ以上のノードがダウンすると、ダウンした 1 つ以上のノードのポッドが保留状態になる可能性があります。

  次の条件は、同じゾーンノードと異なるゾーンノードに適用されます。

  - 同じゾーンノード：同じゾーン内のノードに、そこでポッドをスケジュールするために要求された CPU またはメモリリソースがない。
  - 異なるゾーンノード：次の条件のいずれかが異なるゾーンノードに適用されます。
    - 別のゾーン内のノードに、そこでポッドをスケジュールするために要求された CPU またはメモリリソースがない。
    - 別のゾーンのノードに、ポッドをスケジュールするために要求された CPU またはメモリリソースがある。スケジュール設定では、ポッドは maxSkew:1 の制約に違反します。

CDL のオーバーロードの防止

CDL は、システムがフルキャパシティに達するたびに、エンドポイントでオーバーロードを防止します。エンドポイントでのオーバーロード防止により、CDL は次のことができます。

- ソフト制限の上限に達した場合、作成要求を拒否する。

- ハード制限の上限に達した場合、更新 要求を拒否する。
- システムが最大キャパシティの80%および90%に達したときにアラームをトリガーする。
- エンドポイントでオーバーロード防止を設定（有効または無効）する。

CDL の最大キャパシティ

次のパラメータを使用して、CDL のシャードあたりの最大キャパシティを定義できます。

- *slotRecordCapacity*（デフォルト値：1M/シャード）
- *indexRecordCapacity*（デフォルト値：10M/シャード）
- *slotRecordsSizeCapacityInBytes*（デフォルト値：5GB/シャード）

（注） シャードあたりのデフォルトのキャパシティは変更できません。

CDL は次の計算に基づいて最大キャパシティを決定します。

ソフト制限

1. `cdl_slot_record_capacity  =  No of shards * slotRecordCapacity`
2. `cdl_index_record_capacity = No of shards * indexRecordCapacity`
3. `cdl_slot_size_capacity    = No of shards * slotRecordsSizeCapacityInBytes`

ハード制限

1. `cdl_slot_record_capacity = No of shards * slotRecordCapacity * 105%`
2. `cdl_index_record_capacity = No of shards * indexRecordCapacity * 105%`
3. `cdl_slot_size_capacity = No of shards * slotRecordsSizeCapacityInBytes * 105%`

CDL でのオーバーロードのアラートのトリガー

システムが最大キャパシティに達するたびに、CDL は過負荷となります。また、CDL は次のアラートをトリガーして、システムを過負荷から保護します。

- *cdlOverloaded - major*：このアラートは、CDL システムがキャパシティの 80% に達するとトリガーされます。
- *cdlOverloaded - critical*：このアラートは、CDL システムがキャパシティの90%に達するとトリガーされます。

アラートに加え、CDLエンドポイントは次のエラーによってすべての作成要求を拒否します。

```
error code 507: Datastore reached its full capacity
```

ただしCDLは、ハード制限キャパシティに達するまで、すべての更新要求を処理します。ハード制限キャパシティを超えると、CDL はすべての更新要求を拒否します。

オーバーロードの防止の設定

既存の CDL のオーバーロード防止設定は、新しい設定に置き換えられます。

（注） 以前はオーバーロードの防止がデフォルトで有効になっていて、スロットあたり 100 万レコード、インデックスあたり 1,000 万レコード、かつスロットあたり 5 GB のレコードサイズというハードコーディング制限がありました。しかし本リリースではこの機能はデフォルトで無効になっているため、この機能を有効に設定し、各パラメータに適切なオーバーロード制限（オプション）を設定する必要があります。

表 *3: CDL* のオーバーロードデータの防止

コマンド	変更内容
古いコマンド： **cdl datastore session overload-protection disable** *true*	古いCDL オーバーロード防止コマンドは廃止されます。 （注） コマンドは機能しませんが、後方互換性のためにのみ使用できます。
新しいコマンド： **cdl datastore session features overload-protection enable** *<true/false>*	オーバーロード防止の設定は、オーバーロード防止とアラートを設定する **cdl データストアセッション機能**の設定の下に移動されます。 CDL では現在、次のものを設定することが可能です。 • スロット/インデックスあたりのレコードキャパシティ • スロットあたりのレコードキャパシティ（バイト単位） • メジャーおよびクリティカルアラームの設定（%）

オーバーロード防止が有効になっている場合は、アラートも有効になります。prometheus-rules-cdl ポッドが生成されます。オーバーロード防止が無効になっている場合、アラートは無効になります。prometheus-rules-cdl ポッドが削除されます。

オーバーロード防止パラメータの設定

オーバーロード防止の制限を設定するために、次のパラメータが設定されています。

- **cdl datastore session features overload-protection index-max-record-count** *<value>*
- **cdl datastore session features overload-protection slot-max-record-count** *<value>*
- **cdl datastore session features overload-protection slot-max-size** *<value>*
- **cdl datastore session features overload-protection hard-limit-percentage** *<value>*

次の表に、設定の詳細を示します。

表 *4*:オーバーロード防止パラメータの設定

CLI コマンド	説明
cdl datastore session features overload-protection enable *<true/false>*	（オプション）CDL オーバーロード防止はデフォルトでは無効になっています。デフォルト値は false です。
cdl datastore session features overload-protection index-max-record-count *<value>*	（オプション）インデックスシャードに保存できるレコードの最大数。 デフォルト値は 60000000（60M）です。 指定できる範囲は 100k ー 100M です。 （注） 100 ～ 1000 の範囲は、ラボ環境でのテストにのみ適用されます。実稼働環境での適用はお勧めしません。
cdl datastore session features overload-protection slot-max-record-count *<value>*	（オプション）スロットシャードに保存できるレコードの最大数。 デフォルト値は 2500000（2.5M）です。 範囲は 100 または 100k ～ 10M のいずれかです。 （注） 100 の値は、ラボ環境でのテストにのみ適用されます。実稼働環境での適用はお勧めしません。

CLI コマンド	説明
cdl datastore session features overload-protection slot-max-size *<value>*	（オプション）スロットシャードの最大サイズ（メガバイト単位）。 デフォルト値は 16384（16GB）です。 範囲は 1GB ～ 96GB です。
cdl datastore session features overload-protection hard-limit-percentage *<value>*	（オプション）ソフト制限に加える、追加のキャパシティ（パーセント）。これは、CDL エンドポイントで更新要求を拒否する場合を判断するために使用されます。たとえば、index shard = 1、index-record-capacity = 100、hard-limit-percentage = 5 の場合、インデックスレコードの数が 100 のときは作成要求が拒否され、更新要求は 105 に達した場合にのみ拒否されます。 デフォルト値は 5 です。指定できる範囲は 0 ～ 10 です。
cdl datastore session features overload-protection major-alert-threshold *<value>*	（オプション）CDL がアラート cdlOverloaded-major をトリガーするしきい値（パーセンテージ）。 デフォルト値は 80 です。 指定できる範囲は 40 ～ 100 です。
cdl datastore session features overload-protection critical-alert-threshold *<value>*	（オプション）CDL がアラート *cdlOverloaded-critical* をトリガーするしきい値（パーセンテージ）。 デフォルトは 90 です。 指定できる範囲は 40 ～ 100 です。

アラートのパーセンテージを設定するには、次のコマンドを実行します。

cdl datastore session features overload-protection critical-alert-threshold *<percentage>*

cdl datastore session features overload-protection major-alert-threshold *<percentage>*

アラートの確認

CEE Ops Center CLI で次のコマンドを使用して、アラートを確認できます。

show alerts active { **detail** | **summary** }

また、次のコマンドを使用して、CEE Ops Center CLI でアラートをフィルタ処理できます。

```
show alerts active detail sumary "CDL is overloaded."
```

次の例では、システムキャパシティが 80% 以上になった場合に *cdlOverloaded - major* アラートがトリガーされます。

```
alerts active detail cdlOverloaded-major 5446095ab264
 severity    major
 type        "Processing Error Alarm"
 startsAt    2020-10-15T15:09:00.425Z
 source      System
 summary     "CDL is overloaded."
```

次の例では、システムキャパシティが 90% 以上になった場合に *cdlOverloaded - critical* アラートがトリガーされます。

```
alerts active detail cdlOverloaded-critical 5446095ab264
 severity    critical
 type        "Processing Error Alarm"
 startsAt    2020-10-15T15:09:16.425Z
 source      System
 summary     "CDL is overloaded."
```

CDL のラック変換

機能説明

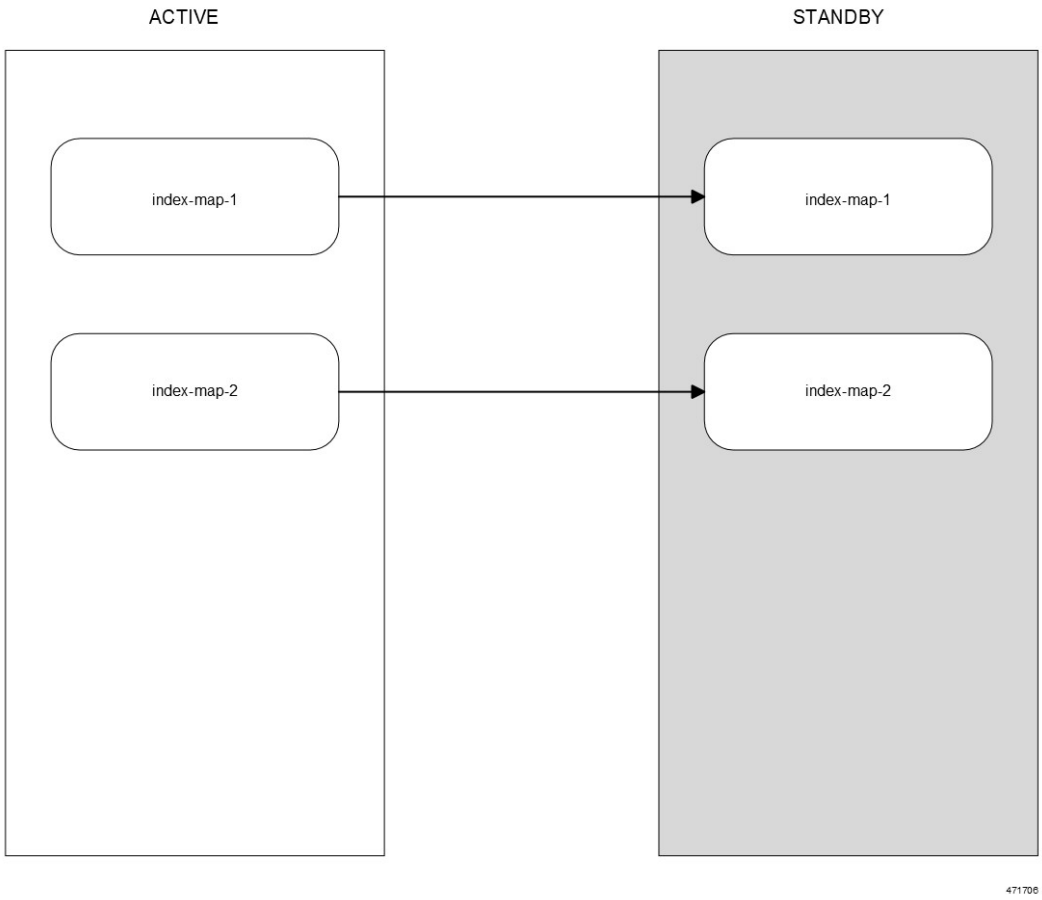
CDLは、再調整メソッドを使用したトラフィックのドレインなしでの、ハーフハイトからフルハイトへのラック変換をサポートしています。

この再調整メソッドは、移行中の既存のコールと新しいコールには影響しません。この機能は、CDL のインデックスマップの再調整に対応します。

CDL 変換の MOP

CDL のハーフラック構成をフルラック構成に移行するには、次の手順を使用します。

1. ラック 1 をアクティブ、ラック 2 をスタンバイに設定します。

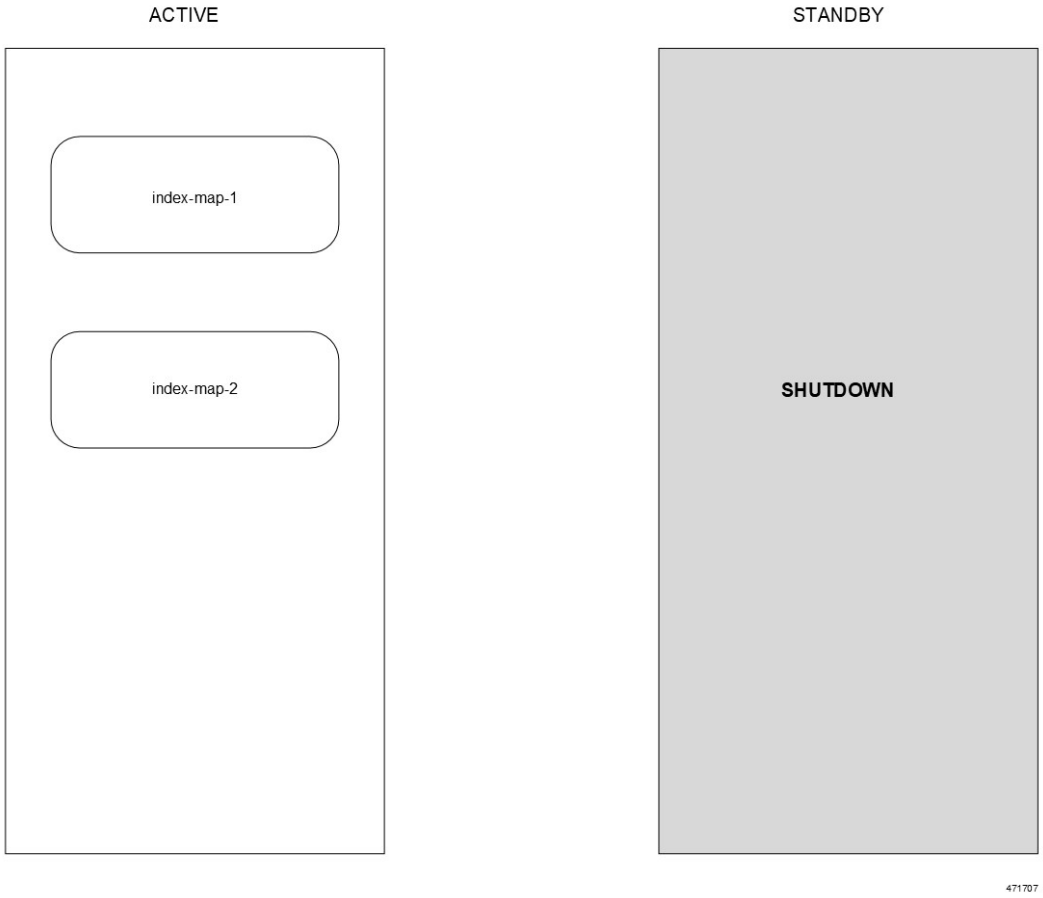


前提条件：*CDL* はハーフラック構成を使用して実行されていますが、両方のサイトにフルラックのリソースを備える必要があります。

アクション：ラック *1* をアクティブにし、ラック *2* をスタンバイにします。*CDL* の設定変更はありません。

事後検証：トラフィックはラック *1* にのみ到達します。

2. ラック 2 でシステム モード シャットダウンを設定します。



前提条件：ラック *1* はアクティブモード、ラック *2* はスタンバイモードである必要があります。

アクション：ラック *2* でシステム モード シャットダウンを実行します。

事後検証：サイト *1* の *mirror-maker* は実行状態ではありません。

3. ラック 2 をスタンバイに設定します。新しい設定に合わせてフルラック CDL を構成し、prev-map-count が設定されたスケールアップモードを有効にします。

   以下に設定例を示します。

```
configure
   cdl datastore session mode scale-up
   cdl datastore session index prev-map-count 2
```

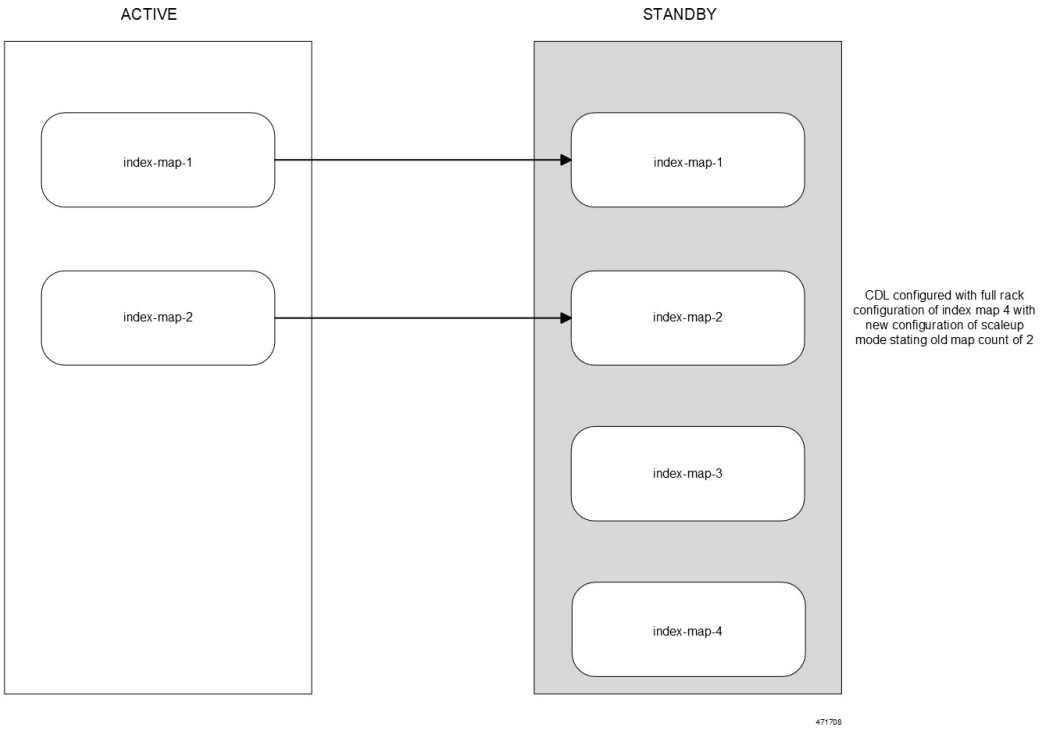
   スケールアップモードなしのハーフラック構成：

```
cdl datastore session
label-config session
   index map 2
   slot map 4
   exit
```

スケールアップモードを使用するフルラック構成：

```
cdl datastore session
label-config session
mode scale-up
   index map 4
   index prev-map-count 2
   slot map 8
   exit
```

4. ラック 2 の system mode running を有効にします。

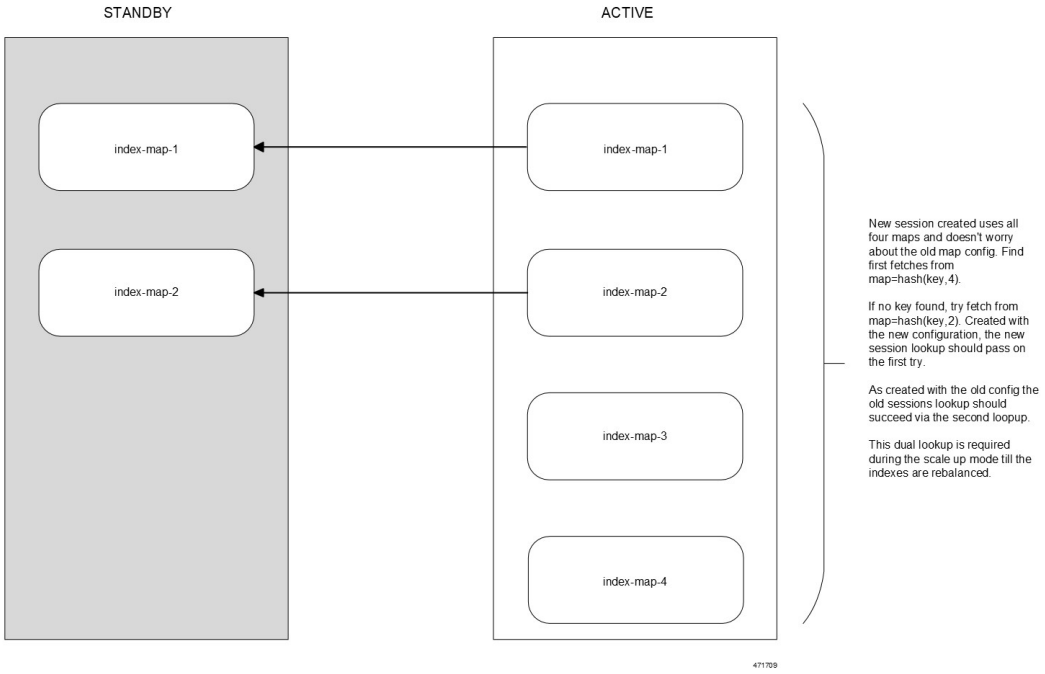


前提条件：ラック *1* はアクティブである必要があり、ラック *2* はフルラック *CDL* 構成でスタンバイである必要があります。

アクション：ラック *2* で *system mode running* を実行します。

事後検証：両方のサイトの *mirror-maker* が実行状態である必要があります。サイト *2* の古いインデックスまたはスロットが、リモートピアとの初期同期を実行できる必要があります。

5. トラフィックをラック 2 に切り替えます。

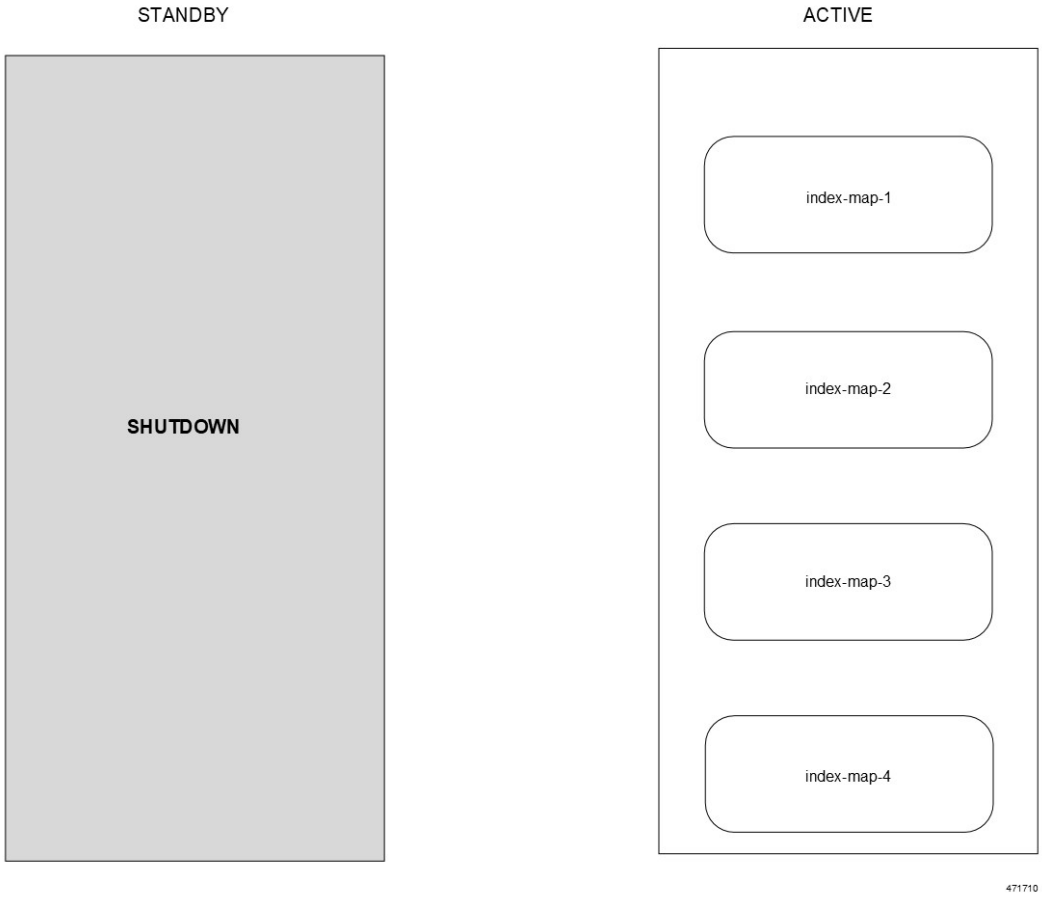


前提条件：ラック *1* はアクティブになっており、ハーフラック構成で実行されている *CDL* を含むフルラックリソースを備えている必要があります。ラック *2* はスタンバイになっており、フルラック構成で実行されている *CDL* を含むフルラックリソースを備えている必要があります。

アクション：ラック *1* をスタンバイにし、ラック *2* をアクティブにします。*CDL* で設定を変更する必要はありません。

事後検証：トラフィックはラック *2* にのみ到達します。

6. ラック 1 のシステム モード シャットダウンを有効にします。



前提条件：ラック *2* はアクティブ、ラック *1* はスタンバイである必要があります。

アクション：ラック *1* でシステム モード シャットダウンを実行します。

事後検証：サイト *2* の *mirror-maker* は実行状態ではありません。

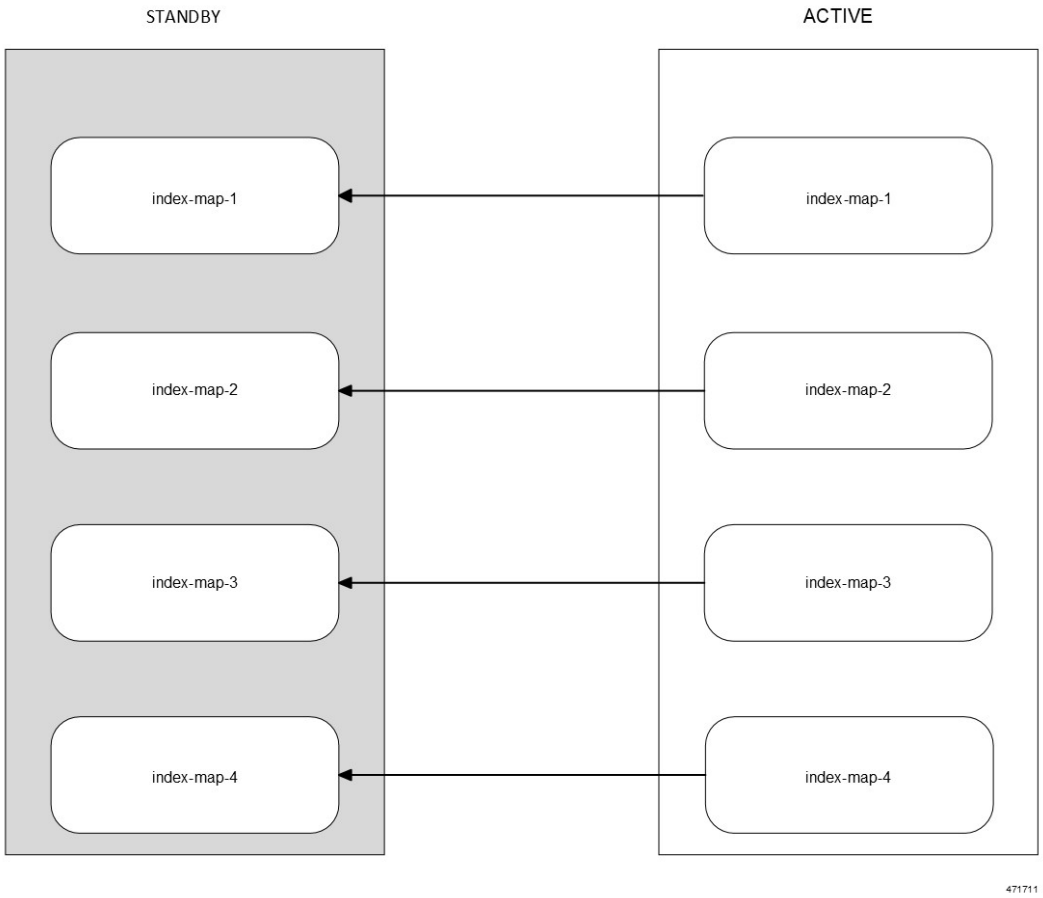
7. ラック 1 をスタンバイに設定します。新しい設定でフルラック CDL を構成し、premap-count が設定されたスケールアップモードを有効にします。

   以下に設定例を示します。

   ```
   configure
     cdl datastore session mode scale-up
     cdl datastore session index prev-map-count 2
   ```

   フル構成により、スロットマップ数とインデックスマップ数が更新されます。新しく追加されたスロットとインデックスのラベル設定が追加されます。

8. ラック 1 の system mode running を有効にします。

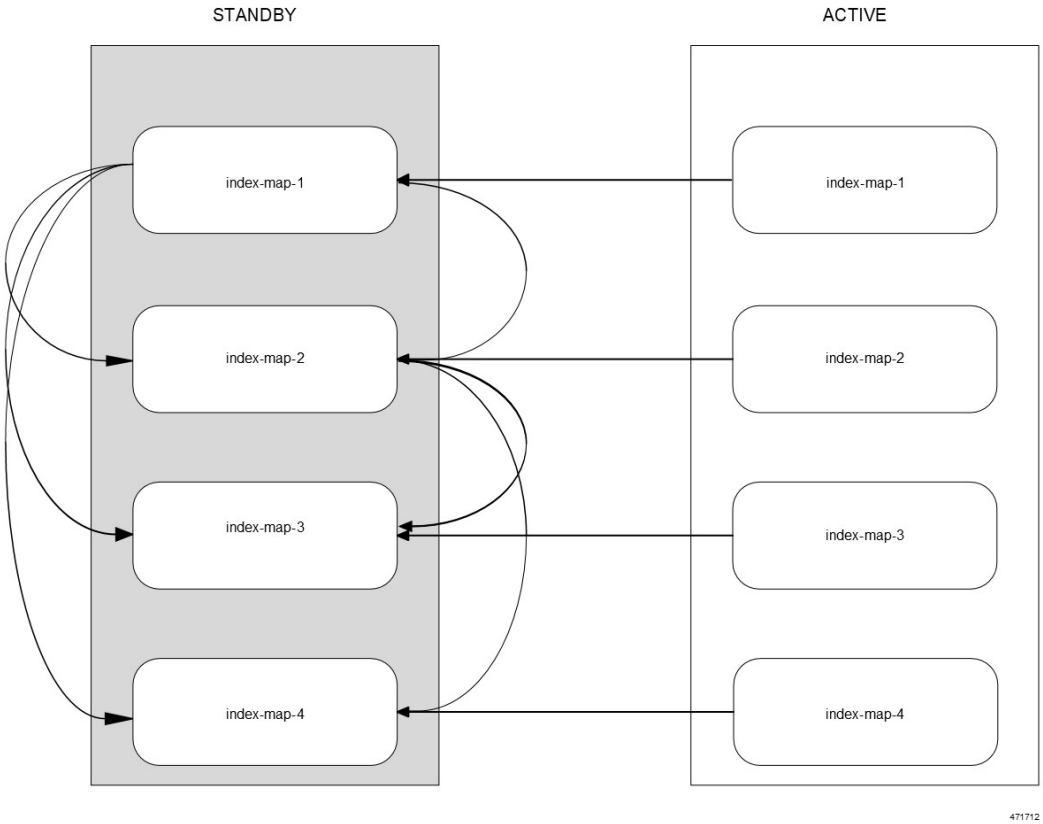


前提条件：ラック *2* はアクティブである必要があり、ラック *1* はフルラック *CDL* 構成でスタンバイである必要があります。

アクション：ラック *1* で *system mode running* を実行します。

事後検証：両方のサイトの *mirror-maker* が実行状態である必要があります。サイト *1* のすべてのインデックスとスロットが、リモートピアとの初期同期を実行できる必要があります。*CDL* レプリケーションエラーがあるかどうかを *CEE* アラートで確認します。

9. ラック 1（スタンバイ）の CLI を使用してインデックスの再調整をトリガーします。



前提条件：*CDL* が両方のサイトで正常な状態になっている必要があります。ラック *1* はスタンバイ、ラック *2* はアクティブである必要があります。両方のサイトに、スケールアップモードがインデックス *prev-map-count* で設定されている必要があります。

ラック *1* で *CLI* によるインデックスの再調整を行うと、（古いマップ数に応じて）以前計算されたマップから（新しいマップ数に応じて）新しく計算されたマップにインデックスがコピーされます。これは、コピーが完了し、以前計算されたマップからインデックスを削除した後に行われます。

アクション：スタンバイサイト（ラック *1*）で次のコマンドを使用して、*rebalance-index* の実行をトリガーします。

cdl rebalance-index run tps *tps_value*

事後検証：再調整ステータスを追跡するには、次の手順を実行します。

- 次の CLI コマンドを使用して、再調整ステータスをモニターします。

  cdl rebalance-index status

- 次の CLI コマンドを使用して、再調整の実行を検証します。

  cdl rebalance-index validate

10. インスタンスをアクティブ-アクティブに切り替えます。

前提条件：再調整が正常に完了する必要があります。

アクション：ラック *1* とラック *2* をアクティブとして有効にします。

事後検証：トラフィックが両方のサイトに到達する必要があります。

11. 両方のサイトから CDL スケールアップモード設定を削除します。

前提条件：ラック *1* とラック *2* の両方がアクティブである必要があります。

アクション：両方のサイトで次のコマンドを設定します。

```
no cdl datastore session mode
```

事後検証：*CDL* が正常な状態で実行されている必要があります。スケールアップモードが無効になると、*mirror-maker* が再起動します。

トラブルシューティング

再調整ログを表示するには、CLI で次のロガーを設定します。

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

再調整されたインデックスキーの総数を把握するために、インデックスポッドに次のメトリックが導入されています。

メトリック	ラベル
index_rebalanced_keys_total	cdl_slice、shardId

CDL 用の FindAll および FindAllNotify クエリの機能拡張

機能説明

このリリースでは、CDL は FindAll クエリと FindAllNotify クエリの拡張機能をサポートしています。

次の操作をサポートするために、新しいクエリフィルタが導入されています。

- FindAllNotify クエリに最大 20 の IN フィルタ。CDL はスロットからすべてのセッションを取得した後、各セッションを渡されたフィルタと照合します。セッションキーが渡されたフィルタのいずれかと一致した場合、一致した IN フィルタ条件とともに NF に通知します。
- 複数の条件を持つ NOT-MATCH 操作。
- CDL CLI の IN 操作と NOT 操作。
- 通知の IPC ストリーミング。

  以前のリリースでは、CDL はアプリケーションへの通知の送信に単項 RPC を使用していました。このリリースでは、CDL は自身とアプリケーション間の IPC を最適化するために、双方向ストリーミング RPC をサポートしています。

機能の仕組み

このセクションでは、FindAll クエリと FindAllNotify クエリでさまざまなフィルタ条件をサポートする、CDL の動作のシーケンスについて説明します。

IN フィルタのサポート

1. システムは、クエリパラメータ内のすべての AND 条件と IN 条件をチェックします。
   - クエリに AND と IN の両方のクエリフィルタが含まれている場合は、次の方法で処理されます。

- CDL は、まずすべての AND フィルタをチェックします。いずれかの AND フィルタが失敗した場合、そのキーはフィルタに一致しなかったので、他の AND フィルタと IN フィルタはスキップされます。
- すべての AND フィルタが一致したら、IN フィルタをチェックします。いずれかの IN フィルタが一致した場合、そのキーはすでにフィルタに一致しているため、CDL は他の IN フィルタをスキップします。

- クエリに IN フィルタのみが存在し、いずれかの IN フィルタが一致した場合、CDL は他の IN 条件の処理をスキップします。

2. クエリ要求に 20 を超える IN フィルタがある場合は、400 Bad Request メッセージで応答を返します。
3. CDL は、FindAllNotify クエリ要求で有効になっている場合にのみ、通知とともに一致した IN フィルタ条件を送信します。

（注） 要求でより多くの IN フィルタが渡されると、クエリのパフォーマンスに影響します。

複数の条件での NOT-MATCH 操作のサポート

CDL は、このクエリのフィルタ処理で次の追加条件をサポートしています。

- not-match
- not-starts-with
- not-ends-with
- not-contains

（注） セッションのいずれかのキー（pk/uk/nuk）が **match**、**contains**、**starts-with**、または **ends-with** の 条件を満たす場合、そのセッションは **not-match**、**not-contains**、**not-starts-with**、または **not-ends-with** の条件のそれぞれに対応するフィルタには選択されません。

CDL CLI での IN 操作と NOT 操作のサポート

- CLI でセッションをフィルタ処理するために、次の追加の条件がサポートされています。
  - が次の文字列を含む（contains）
  - not-contains
  - not-ends-with
  - not-starts-with

- not-match

- CLI での IN 操作では、AND フィルタ **filter** に加えて、**in-filter** フィルタもサポートされます。

  次のコードスニペットは、CLI コマンドの例です。

```
cdl show sessions summary filter { condition ends-with key 6 } in-filter { in-condition
 starts-with key key-1 } in-filter { in-condition starts-with key key-2 }
```

（注） **filter** および **in-filter** フィルタは、条件のリストです。クエリは最大 20 の **in-filter** フィルタをサポートします。**filter** で言及されているフィルタはグループ化されているため、フィルタ、**filter**、および **in-filter** の順序は重要ではありません。同様に、**in-filter** で言及されているフィルタもグループ化されています。

- **filter** と **in-filter** によるフィルタ処理では、次の CDL 操作がサポートされています。
  - **cdl show sessions count summary**
  - **cdl show sessions count detailed**
  - **cdl show sessions detailed**
  - **cdl show sessions summary**
  - **cdl clear sessions**

ストリーミング通知のサポート

- CDL は、双方向ストリーミング RPC を使用して、トランザクション ID を持つストリームを介して通知をアプリケーションに送信します。応答は、関連付けのためのトランザクション ID とともに、アプリケーションからデータストアに異なるストリームで送信されます。
- デフォルトでは、CDL は単項 RPC を使用し、ストリーミング機能は無効になっています。ストリーミングは、機能フラグ **use-stream** が CDL CLI で設定されている場合にのみ有効になります。ストリーミングが有効になっている場合、CDL 通知はすべてストリーミングを使用して送信されます。

  通知のストリーミング機能を有効にするには、次の CLI コマンドを使用します。

```
cdl datastore session slot notification use-stream true
```

- ストリーミングが有効になっている場合、デフォルトでは、CDL から通知エンドポイントへ 4 つのストリーム接続があります。接続数は CDL CLI を使用して設定できます。ストリーム接続はラウンドロビン方式で使用されます。

- スロットまたはエンドポイントから通知エンドポイントへのアクティブなストリーム接続の数を確認するためのメトリックをサポートします。これらのメトリックを使用して、アクティブなストリーム接続がない場合にアラートを生成します。

ネットワーク ポリシー設定

機能の概要と変更履歴

要約データ

該当製品または機能エリア	Cloud Native Broadband Network Gateway（cnBNG）2022.02 以降
該当するプラットフォーム	ベアメタル、OpenStack、VMware
機能のデフォルト設定	無効：設定が必要
このリリースでの関連する変更点	N/A
関連資料	*UCC CDL Configuration and Administration Guide* リリース *1.8*

マニュアルの変更履歴

改訂の詳細	リリース
最初の導入。	CDL 1.8.0

機能説明

このリリースでは、CDL を使用して、ネットワークポリシーを設定しグローバル ネットワーク ポリシー設定をオーバーライドすることができます。設定が有効になっている場合、CDL エンドポイント、インデックス、スロット、Kafka、およびZookeeperのネットワークポリシーが適用されます。

etcd のネットワークポリシーを設定して、グローバル ネットワーク ポリシー設定をオーバーライドすることもできます。設定が有効になっている場合、etcdのネットワークポリシーが適用されます。

CDL ネットワークポリシーの設定

CDL ネットワークポリシーを設定するには、次の CLI コマンドを使用します。

```
config
  cdl network-policy enable <true/false>
```

etcd ネットワークポリシーを設定するには、次の CLI コマンドを使用します。

```
config
  etcd network-policy enable <true/false>
```

トラブルシューティング情報

このセクションでは、CDL 設定に関する既知の問題の一部を解決する方法について説明します。CDL 設定の障害対応に進む前に、次の点を考慮してください。

- **system-id** は、サイト全体で一意です。
- **cluster-id**（**datastore id**）は、サイト内で一意です。
- CDLの展開後は、インデックスのレプリカを変更できません。レプリカを変更するには、次の手順を実行します。
  1. システムモードを「シャットダウン」に設定します。
  2. レプリカを変更します。
  3. システムモードを「実行」に設定します。
- エンドポイント、インデックス、スロット、およびKafkaポッドのレプリカは、次のラベルと値を持つ k8 ノードの数以下である必要があります。
  - ラベル：*smi.cisco.com/node-type*
  - 値：*cdl/node-type* の値マルチノード設定の場合、デフォルト値は *session* です。
- ZookeeperおよびETCDポッドのレプリカは、次のラベルと値を持つk8ノードの数以下である必要があります。
  - ラベル：*smi.cisco.com/node-type*
  - 値：*oam*
- Geo レプリケーションのセットアップ（サイトの cluster-id 間）では、レプリカと、インデックスおよびスロットポッドのマップを同一にする必要があります。そうしないと、データのレプリケーションが失敗する可能性があります。

CDL インデックスが正確にレプリケートされない

このセクションでは、CDLインデックスポッドを正確にレプリケートする方法について説明します。

問題の説明

セッションデータがリモートサイトに正しくレプリケートされない。

問題の特定

一方のサイトで追加されたデータに、もう一方のサイトではアクセスできません。

考えられる原因

不適切な Geo レプリケーションの設定がこの問題の原因となっている可能性があります。

対処法

この問題を解決する手順は、次のとおりです。

- ローカルの system-id とリモートサイトの設定を確認します。
- 各サイト間の CDL エンドポイントと Kafka が到達可能かどうかを確認します。
- 各サイトでマップ、インデックスのレプリカ、およびスロットを確認します。それらはすべてのサイトで同一である必要があります。

CDL の操作が失敗しているのに、接続は成功する

このセクションでは、CDL 操作の失敗の問題を解決する方法について説明します。

問題の説明

NF は CDL に接続しているが、検索、作成、更新、削除などのセッション操作は失敗する。

問題の特定

NF ログファイルを確認すると、コールの失敗を特定できます。

考えられる原因

この問題は、CDLインデックスとスロットポッドの準備ができていない場合に発生する可能性があります。

対処法

この問題を解決する手順は、次のとおりです。

- すべてのポッドの準備ができており、実行状態であるかどうかを確認します。
- インデックスポッドは、ピアレプリカ（使用可能な場合はローカルまたはリモート）との同期が完了した場合にのみ、準備完了状態に移行します。
- スロットポッドは、ピアレプリカ（使用可能な場合はローカルまたはリモート）との同期が完了した場合にのみ、準備完了状態に移行します。

- エンドポイントを準備完了状態に移行するには、少なくとも 1 つのスロットとインデックスポッドが使用可能である必要があります。準備ができていない場合でも、クライアントは GRPC 接続を受け入れます。

CDL のポッドがダウンしている

このセクションでは、CDL ポッドがダウンしている場合にこれらを起動する方法について説明します。

問題の説明

CDL の設定が正しくないため、CDL ポッドが「実行」状態にならない。

問題の特定

次のコマンドを使用して、「describe pods」の出力（コンテナ、メンバー、状態、理由、またはイベント）を確認し、ポッドがダウンしているかどうかを識別します。

kubectl describe pods -n *<namespace> <failed pod name>*

考えられる原因

考えられる原因は次のとおりです。

- ポッドが「*pending*」の状態である。
- ポッドが「*CrashLoopBackOff*」障害の状態である。
- ポッドが「*ImagePullBack*」障害の状態である。

対処法

- ポッドが「*pending*」状態の場合：
  - ラベル値 *cdl/node-type* を持つ k8s ノードが存在するかどうかを確認します。また、レプリカの数が、ラベル値 *cdl/node-type* を持つ k8s ノードの数以下であることも確認します。

    ```
    kubectl get nodes -l smi.cisco.com/node-type=<value of cdl/node-type, default
    value is 'session' in multi node setup)
    ```

- ポッドが「*CrashLoopBackOff*」障害の状態の場合：
  - ETCD ポッドのステータスを確認します。

    ```
    kubectl describe pods -n <namespace> <etcd pod name>
    ```

    ETCD ポッドが実行されていない場合は、ETCD の問題を解決してポッドを起動します。

- ポッドが「*ImagePullBack*」障害の状態の場合：

- helm リポジトリとイメージレジストリにアクセスできるかどうかを確認します。
- 必要なプロキシサーバーと DNS サーバーが設定されているかどうかを確認します。

準備中の状態の Mirror Maker ポッド

このセクションでは、「準備中（*Not Ready*）」の状態のままの Mirror Maker ポッドを移動する方法について説明します。

問題の説明

リモートサイトでの接続の問題により、Mirror Maker ポッドが実行状態ではない。

問題の特定

次のコマンドを使用して、Describe ポッドの出力およびポッドログを確認します。

```
kubectl describe pods -n <namespace> <failed pod name>
kubectl logs -n <namespace> <failed pod name> [-c <container name>]
```

考えられる原因

リモートサイトの Kafka ブローカとの接続の問題がこの問題の原因となっている可能性があります。

対処法

この問題を解決する手順は、次のとおりです。

- Kafka 用に設定された外部 IP が正確であるかどうかを確認します。
- 外部 IP を介して Kafka のリモートサイトに到達できるかどうかを確認します。

CDL からのレコード消去の通知が早い、または遅延する

このセクションでは、CDL からのレコード消去について送信される通知が早い、または遅延する問題を解決する方法について説明します。

問題の説明

CDL からのレコード消去について送信される通知が早い、または遅延する。

問題の特定

CDL からのレコードの消去中に来る通知が早い、または遅延します。

考えられる原因

Kubernetes クラスタ内のノードの時刻が同期されていない。

対処法

この問題を解決する手順は、次のとおりです。

- k8s クラスタ内のすべてのノードの時刻が同期されているかどうかを確認する。
- すべての k8s ノードの同期ステータスを確認する。

```
chronyc tracking
chronyc sources -v
chronyc sourcestats -v
```

CDL でセッションが蓄積する、またはオーバーロードする

このセクションでは、CDL でのセッションの蓄積を解決する方法について説明します。

問題の説明

ネットワークの問題または CDL 上の想定外のセッション数によりセッション数が蓄積し、ピアセッション数と一致しない。

問題の特定

CLI または Grafana ダッシュボードからのセッション数に、予想よりも多くのセッション数が表示されているか、セッション数が増え続けています。

考えられる原因

CDL で受信した *Create* 要求が *Delete* 要求よりも多くなっています。結果として、セッション数は増加し続けます。

対処法

この問題を解決するには、各 NF がサブスクライバのクリアをトリガーするための適切なアクションを実行する必要があります。

ユーザー定義のスケジュール設定とポッドの配置

このセクションでは、CDL ポッドでのスケジュール設定、ノードアフィニティ、およびポッド配置に単一のノードラベルを使用することによって発生する可能性のある、スケジュール不可のポッドの問題を解決する方法について説明します。

問題の説明

nodeType パラメータは、CDL でのスケジュール設定、ノードアフィニティ、およびポッド配置を制御する。現在、このパラメータは単一のノードラベル値 *smi.cisco.com/node-type* で設定されている。CDL がフルノードキャパシティに設定されている場合、単一のノードラベルを使用することによって、（場合により）ポッドがスケジュール不可とマークされる。

問題の特定

CDL ポッドがスケジュール不可とマークされます。

考えられる原因

スケジュール設定、ノードアフィニティ、およびポッド配置に単一のノードラベルを使用している。

対処法

個別のノードラベルを使用して、さまざまな CDL ポッドタイプを制御できます。個別のノードラベルを作成するには、次の手順を実行します。

1. （必須）SMI Cluster Manager Ops Center CLI を介して CDL に使用するすべてのノードで *smi.cisco.com/node-type* ラベルを作成します。
2. 次の表で指定されているように、CDL ポッドのラベルを作成します。

ポッド	ラベル	値
CDL エンドポイントポッド	*smi.cisco.com/cdl-ep*	*true*
CDL スロットポッド	*smi.cisco.com/cdl-slot-<MAP Id 1..n>*	*true*
CDL インデックスポッド	*smi.cisco.com/cdl-index-<MAP Id 1..n>*	*true*

（注） この設定は、ユーザー定義のスケジュール設定にのみ必要です。

例

次の例は、2 つのエンドポイントレプリカ、4 つのスロットマップ（それぞれ 2 つのレプリカ）、および 2 つのインデックスマップ（それぞれ 2 つのレプリカ）を持つ 4 つの異なるノードを使用する CDL に個別のノードラベルを作成する方法を示しています。

ノード 1	ノード 2	ノード 3	ノード 4
ポッド：slot-1-0	ポッド：slot-1-1	ポッド：slot-2-0	ポッド：slot-2-1
ラベル：cdl-slot-1	ラベル：cdl-slot-1	ラベル：cdl-slot-2	ラベル：cdl-slot-2
ポッド：slot-3-0	ポッド：slot-3-1	ポッド：slot-4-0	ポッド：slot-4-1
ラベル：cdl-slot-3	ラベル：cdl-slot-3	ラベル：cdl-slot-4	ラベル：cdl-slot-4
ポッド：index-1-0	ポッド：index-1-1	ポッド：index-2-0	ポッド： index-2-1

ノード 1	ノード 2	ノード 3	ノード 4
ラベル： cdl-index-1	ラベル： cdl-index-1	ラベル： cdl-index-2	ラベル： cdl-index-2
ポッド：ep-0			ポッド：ep-2
ラベル：cdl-ep	ラベル：cdl-ep	ラベル：cdl-ep	ラベル：cdl-ep

設定例：SMI Cluster Manager

```
clusters user-k8s
  nodes session1
   k8s node-labels smi.cisco.com/cdl-ep true
   exit
   k8s node-labels smi.cisco.com/cdl-index-1 true
   exit
   k8s node-labels smi.cisco.com/cdl-slot-1 true
   exit
   k8s node-labels smi.cisco.com/cdl-slot-3 true
   exit
   k8s node-labels smi.cisco.com/node-type session
   exit
  exit
exit
clusters user-k8s
  nodes session2
   k8s node-labels smi.cisco.com/cdl-ep true
   exit
   k8s node-labels smi.cisco.com/cdl-index-1 true
   exit
   k8s node-labels smi.cisco.com/cdl-slot-1 true
   exit
   k8s node-labels smi.cisco.com/cdl-slot-3 true
   exit
   k8s node-labels smi.cisco.com/node-type session
   exit
  exit
exit
clusters user-k8s
  nodes session3
   k8s node-labels smi.cisco.com/cdl-ep true
   exit
   k8s node-labels smi.cisco.com/cdl-index-2 true
   exit
   k8s node-labels smi.cisco.com/cdl-slot-2 true
   exit
   k8s node-labels smi.cisco.com/cdl-slot-4 true
   exit
   k8s node-labels smi.cisco.com/node-type session
   exit
  exit
exit
clusters user-k8s
  nodes session3
   k8s node-labels smi.cisco.com/cdl-ep true
   exit
   k8s node-labels smi.cisco.com/cdl-index-2 true
   exit
   k8s node-labels smi.cisco.com/cdl-slot-2 true
   exit
   k8s node-labels smi.cisco.com/cdl-slot-4 true
   exit
```

```
   k8s node-labels smi.cisco.com/node-type session
   exit
  exit
exit
```

サンプル設定：CDL

```
cdl label-config session
  endpoint key smi.cisco.com/cdl-ep
  endpoint value true
  slot map 1
    key smi.cisco.com/cdl-slot-1
    value true
  slot map 2
    key smi.cisco.com/cdl-slot-2
    value true
  slot map 3
    key smi.cisco.com/cdl-slot-3
    value true
  slot map 4
    key smi.cisco.com/cdl-slot-4
    value true
  index map 1
    key smi.cisco.com/cdl-index-1
    value true
  index map 2
    key smi.cisco.com/cdl-index-2
    value true
cdl datastore session
  label-config session
  endpoint replica 2
  index replica 2
  index map 2
  slot replica 2
  slot map 4
exit
```

モニタリング

このセクションでは、CDLのステータスとレコードをモニターする方法について説明します。Grafana ダッシュボードを介して CDL をモニターすることもできます。

Grafana ダッシュボードを介した CDL のモニタリング

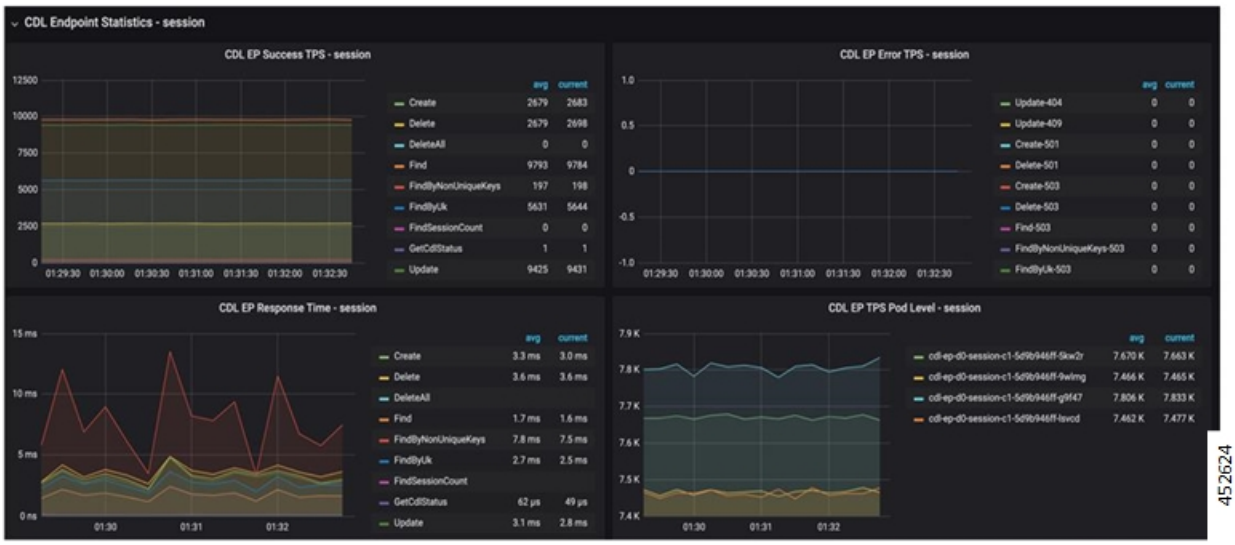
デフォルトでバンドルされているCDLダッシュボードと呼ばれるGrafanaダッシュボードを使用して、CDL のさまざまなアクティビティをモニターできます。CDL ダッシュボードには、*cdl-endpoint*、スロット、インデックス などのさまざまなポッドの CDL に対する TPS と、各操作にかかる応答時間が表示されます。

Type、SliceName、SystemID などで分類した合計レコード数を表示する Grafana CDL Summary ダッシュボードの例を以下に示します。

図 *20 : Grafana CDL* ダッシュボード：概要

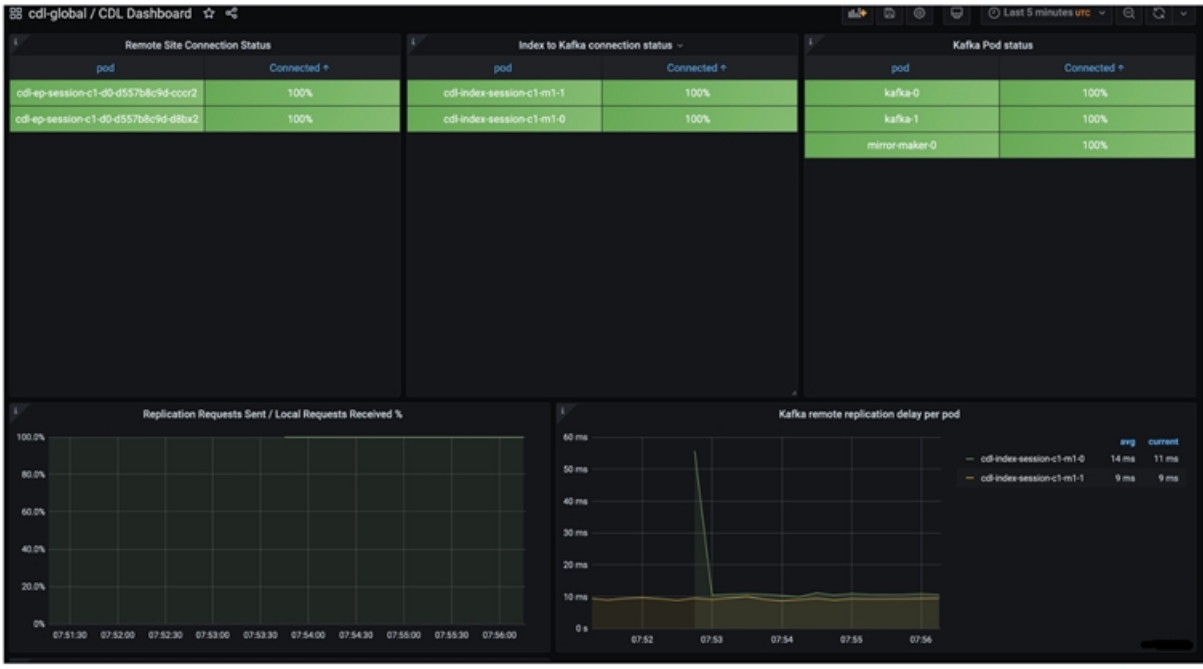
CDL TPS と応答時間のグラフを表示する Grafana CDL ダッシュボードの例を以下に示します。

図 *21 : Grafana CDL* ダッシュボード：*CDL TPS* と応答時間



Grafana CDL ダッシュボードの拡張により、Geo レプリケーションのステータスが表示されます。Geo レプリケーションのステータスとその他の詳細を表示する Grafana ダッシュボードの例を以下に示します。

図 *22 : Grafana CDL* ダッシュボード：*GR* 接続ステータス



GR 接続、インデックス レプリケーションとスロットレプリケーションのパネル、およびそれらの説明を次の表に示します。

表 *5 : GR* 接続ステータス

パネル	説明
リモートサイトの接続ステータス	エンドポイントポッドごとのアクティブなリモートサイト接続（パーセンテージ）。0の状態が 5 分を超えると、アラートがトリガーされます。
インデックスから Kafka への接続ステータス	各インデックスポッドからの現在アクティブな Kafka 接続の平均値。
Kafka ポッドのステータス	Kafka ポッドと mirrorMaker ポッドの準備ステータス。
送信したレプリケーション要求/受信したローカル要求 %	NF が受信したローカル要求に対する、リモートサイトに送信したレプリケーション要求の比率。この比率が 5 分を超えて 90% を下回ると、アラートがトリガーされます。

パネル	説明
ポッドごとの Kafka リモートレプリケーションの遅延	Kafka を介してパートナーサイトにレコードをレプリケートする際の合計遅延。レコードのレプリケートで 10 秒を超える遅延が 5 分を超えて続くと、アラートがトリガーされます。
ドロップされたリモート要求の総数	キューがいっぱいであるためにドロップされたリモート要求の総数。

表 *6:* インデックスのレプリケーション

パネル	説明
ポッドごとの Kafka パブリッシュ TPS	Kafka 書き込み要求（パブリッシュ）のインデックスポッドごとの合計レート。
ポッドごとの Kafka リモートレプリケーション TPS	パートナーサイトから受信する Kafka 要求（消費）のインデックスポッドごとの合計レート。

表 *7:* スロットのレプリケーション

パネル	説明
送信したスロット Geo レプリケーション要求	操作ごとの、cdl-ep からリモートサイトへのレプリケーション要求の合計レート。
受信したスロット Geo レプリケーション要求	操作タイプごとの、スロットポッドが受信したレプリケーション要求の合計レート。
スロットチェックサムの不一致	スロットチェックサム不一致の合計レート。
スロットの調整	スロットデータのチェックサムの不一致によるスロット調整の合計レート。

CDL のステータスとレコードの確認

CDL Ops Center で **cdl show status** コマンドを使用して、CDL のステータスを確認し、クライアントによって挿入されたデータを調べることができます。**cdl show status** コマンドの詳細については、『*UCC CDL Command Reference Guide*』を参照してください。

次の例は、すべての CDL コンポーネントのステータスとクライアントのデータを示しています。

```
[unknown] cdl# cdl show status
message params: {cmd:status mode:cli dbName:session sessionIn:{mapId:0 limit:10000
filters:[]}}
site {
    system-id 1
```

```
    cluster-id 1
    db-endpoint {
        state STARTED
    }
    slot {
        map-id 1
        instance-id 2
        records 100
        capacity 1000000
        state STARTED
    }
    slot {
        map-id 1
        instance-id 1
        records 100
        capacity 1000000
        state STARTED
    }
    index {
        map-id 1
        instance-id 2
        records 500
        capacity 2500000
        state ONLINE
    }
    index {
        map-id 1
        instance-id 1
        records 500
        capacity 2500000
        state ONLINE
    }
}
[unknown] cdl#
```

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。