Cisco Common Data Layer

Revision history

Table 1: Revision History

Revision Details	Introductory Release
Added the support for failure handling scenarios.	2023.03.0
Added the procedures for configuration and verification of the event trace data in the Cisco Common Data Layer database record.	2021.02.0

Geo redundancy of Cisco Common Data Layer in SMF

The Cisco Common Data Layer (CDL) is a high-performance, next-generation Key-value (KV) data store layer for the Cloud-Native applications. These applications use the CDL as a state management tool with High Availability (HA) and Geo HA functions.

SMF supports the Geo Redundant (GR) version of the CDL. The CDL endpoint pod is the front end for receiving requests from SMF application to the CDL. When the primary CDL endpoint fails, the SMF sends the database service requests to the next highly rated secondary endpoint. If the next rated endpoint is unavailable, then the SMF reattempts the operation on the subsequent endpoint that has the highest rating and so on.

In case of specific errors or failures from CDL to SMF, the retry mechanism is activated for Create, Find, Update, and Delete requests. The number of retries is based on the SLA.

The successful response to the SMF stops the retry process and sends the response to the application.

With this functionality, SMF provides a non-disrupted N7 or Diameter message handling.

Benefits of using CDL

The benefits of using CDL in SMF include:

- Data storage: Enables the storage of event trace data in the CDL database record. The event trace data shows the call flow event for the subscribers.
- Service Continuity: The system's ability to handle retries and redirect requests to available endpoints enables SMF to maintain smooth operations even during failures.

How CDL works in SMF

Key components involved in CDL Datastore Architecture are:

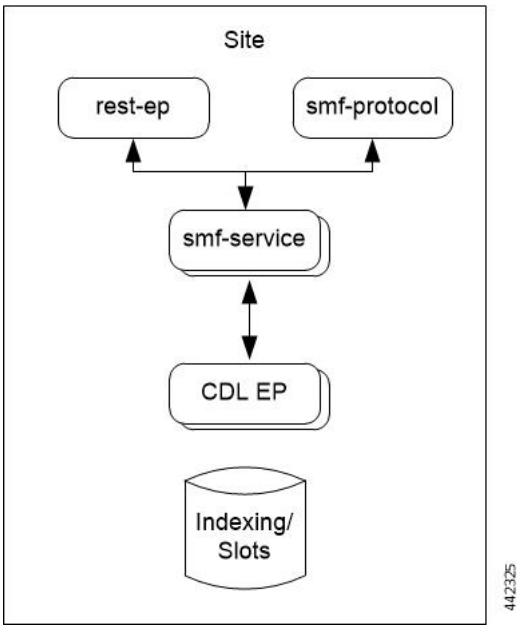
- SMF REST endpoint
- SMF service
- CDL endpoint
- Indexing or slots

Summary

This figure shows the failover process when the SMF service cannot access the CDL datastore endpoint.

Workflow

Figure 1: CDL Datastore Architecture



These stages describe how CDL operates within SMF, along with the failover and error-handling mechanisms.

1. Configuring the CDL through the SMF Ops Center enables the SMF to support multiple CDL datastore endpoints.
2. By default, SMF considers the local endpoint as the primary endpoint, which has the highest rating. SMF performs the CDL API operations on the primary endpoint.
3. If the primary endpoint is unavailable, SMF routes the operations to the next highest-rated endpoint.

   SMF continues to fail over to the next accessible secondary endpoint until all configured secondary endpoints are exhausted. SMF does not reattempt a query on the next rated endpoint if the endpoint is reachable but responds with an error or timeout.

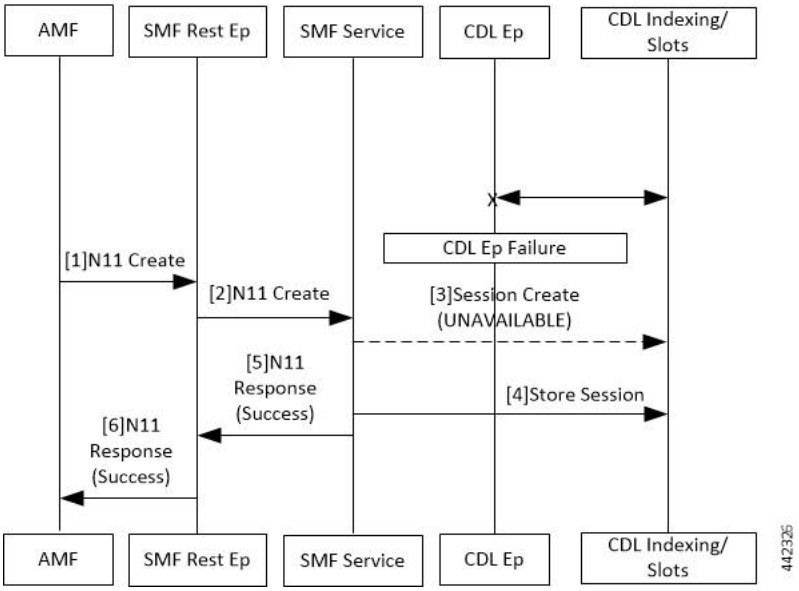
   The CDL operation fails with a 'Datastore Unavailable' error if SMF cannot access any endpoints within the cluster.

   This table provides the when-then condition scenarios for CDL operations.

When	And...	Then the CDL...	And CDL...
all CDL endpoint pods are active and healthy	triggers the CDL operations	does not retry the Create, Read, Update, or Delete operations	do not send any error messages.
a CDL endpoint pod restarts	triggers the CDL operations	retries the Create, Read, Update, or Delete operations	sends the error responses with these error codes: • ERROR_FROM_INDEXING [501] • ERROR_FROM_SLOT [502] • DATASTORE_EP_QUEUE_FULL [503] • MAX_CAPACITY_REACHED [507]

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

Figure 2: CDL endpoint failure call flow



The table details the steps involved in the CDL endpoint failure call flow.

Table 2: CDL endpoint failure call flow description

Step	Description
1	The AMF sends a Create Request to SMF REST endpoint over the N11 interface.
2	After receiving the request, the SMF REST endpoint forwards the Create Request to the SMF service.
3	The SMF service attempts to reach the CDL endpoint to send the session creation request. However, the CDL endpoint is unreachable.
4	The Create Request is evaluated in the stored session. The SMF service then forwards the request to the CDL endpoint.
5	After the call request is successful, the SMF service sends a Success Message to the SMF REST endpoint.
6	The SMF REST endpoint forwards the Success Message to the AMF.

Configure the CDL through SMF Ops Center

The configuration of the CDL using SMF Ops Center involves these steps:

Procedure

Step 1 Configure the CDL session database and define the base configuration, on page 5

Step 2 Configure the Zookeeper in CDL, on page 8

Step 3 Configure event trace data, on page 9

Configure the CDL session database and define the base configuration

Procedure

Step 1 Define the system ID, node type, and zookeeper replica server ID in the global configuration mode to set up the CDL session database.

CLI Command	Description	Range	Default Value
cdl system-id *system_id*	Specifies the system or Kubernetes cluster identity.	NA	1
cdl node-type *node_type*	Specifies the Kubernetes node label to configure the node affinity. The value must be a string.	0–64	session
cdl zookeeper replica *zookeeper_replica_id*	Specifies the zookeeper replica server's ID.	NA	NA

Step 2 Set the log level, cluster, replica, slot, index, and slice parameters to define the base configuration for the CDL datastore session and save the configuration.

CLI Command	Description	Range	Default Value
cdl logging default-log-level *debug_level*	**cdl logging default-log-level** *debug_level*—Specifies the default log level.	NA	NA
cluster-id *cluster-id*	**cluster-id** *cluster-id*—Specifies the Kubernetes cluster identity.	NA	1
endpoint replica *num_replica*	**endpoint replica** *num_replica*—Specifies the number of replicas to be created. The value must be an integer.	*num_replica*—The value must be an integer in the range of 1 to 16.	*num_replica*—The default value is 1.

CLI Command	Description	Range	Default Value
slot { replica *num_replica* | **map** *num_map/shards*| **write-factor** *write_factor* **}**	• **replica** *num_replica*—Specifies the number of replicas to be created. • **map** *num_map/shards*—Specifies the number of partitions in a slot. • **write-factor** *write_factor*—Specifies the number of copies to be written before a successful response. **Note** Make sure that the value is lower than or equal to the number of replicas.	• *num_replica*—The value must be an integer in the range of 1 to 16. • *num_map/shards*—The value must be an integer in the range of 1 to 1024. • *write_factor*—The value must be an integer in the range of 0 to 16.	• *num_replica*—The default value is 1. • *num_map/shards*—The default value is 1. • *write_factor*—The default value is 1.
slot notification { host *host* | **port** *port* | **limit** *tps***}**	• **notification host** *host*—Specifies the notification server hostname or IP address. • **slot notification port** *port*—Specifies the notification server port number. • **slot notification limit** *tps* —Specifies the notification limit per second.	NA	• *host*—The default value is datastore-notification-ep . • *port*—The default value is 8890. • *tps* —The default value is 2000

CLI Command	Description	Range	Default Value
index { replica *num_replica* | **map** *num_map/shards* | **write-factor** *write_factor* **}**	• **replica** *num_replica*—Specifies the number of replicas to be created. • **map** *num_map/shards*—Specifies the number of partitions in a slot. Avoid modifying this value after deploying the CDL. • **write-factor** *write_factor*—Specifies the number of copies to be written before a successful response.	• *num_replica*—The value must be an integer in the range of 1 to 16. • **map** *num_map/shards*—The value must be an integer in the range of 1 to 1024. • **write-factor** *write_factor*—The value must be an integer in the range of 0 to 16.	• *num_replica*—The default value is 2. • *num_map/shards*—The default value is 1. • *write_factor* —The default value is 1.
slice-names *cdl_slice_name*	**slice-names** *cdl_slice_name*—Specify the CDL slice names. The value must be an alphanumeric string.	*cdl_slice_name*—The value must be an integer in the range of 0 to 16.	NA

What to do next

Configure the Zookeeper in CDL

Support for slice-based CDL session statistics

The SMF Support for Slice Based CDL Session Statistics enables the SMF to support session statistics and management commands based on slice parameters, specifically the Slice/Single Network Slice Selection Assistance Information (Snssai) values such as **sst** (Slice/Service Type) and **sd** (Slice Differentiator). Previously, there was no mechanism to determine the number of sessions or perform show and clear operations filtered by these slice parameters.

Sessions are tagged with a CDL non-unique key based on slice values. Key format being, `snssai:<sst><sd>`.

- `sst`: Unsigned integer (0-255), stored as a 3-character string (e.g., 0-9 as "00X", 10-99 as "0XX", 100-255 as "XXX").
- `sd`: 3-octet hexadecimal string representing the Slice Differentiator.

Note **cdl show sessions summary slice-name** command can be used to verify the **snssai** key added in **db_records_total** metrics.

Example query:

```
avg(db_records_total{ session_type=snssai:010,abf123}) by
(session_type,cdl_slice,systemId)
```

Benefits of slice-based CDL session statistics

This section lists the positive outcomes that users can experience from using this feature.

- **Slice-specific session visibility**: Enables operators to view active session counts and details filtered by slice parameters (**sst** and **sd**), improving monitoring granularity
- **Targeted session management**: Allows clearing of subscriber sessions based on slice values, facilitating precise session control and troubleshooting.
- **Integration with CDL metrics**: Leverages CDL statistics and metrics for real-time session counts and verification.
- **Flexible CLI filtering**: Supports exact matches and prefix-based searches (**starts-with**), accommodating various operational needs.

Configure the Zookeeper in CDL

Procedure

Step 1 Specify the size of Zookeeper data storage in gigabyte in the global configuration mode.

cdl zookeeper data-storage-size *data_storage_size_in_gb*

Example:

```
[smf] smf(config)# cdl zookeeper data-storage-size 20 GB
```

Note
The default value is 20 GB. The value must be an integer in the range of 1 to 64.

Step 2 Specify the size of the Zookeeper data log storage in gigabyte.

log-storage-size *log_storage_size_in_gb*

Example:

```
[smf] smf(config-cdl-zookeeper)# log-storage-size 20 GB
```

Note
The default value is 20 GB. The value must be an integer in the range of 1 to 64.

Sample configuration of CDL session database

This sample guides you to configure the Common Data Layer (CDL) session database in SMF.

```
config
cdl system-id system_id
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
 slice-names cdl_slice_name
exit
```

Configure event trace data

This section describes how to configure the SMF to store event trace data in CDL database record. You can enable the storage of event trace data in the CDL database record for system diagnostics. The event trace data shows the call flow event for the subscribers.

Follow these steps to enable the storage of event trace data in the CDL database record:

Procedure

Step 1 Define the system-diagnostics event-trace state to enable in the global configuration mode.

system-diagnostics event-trace *[enable | disable]*

Example:

```
[smf] smf(config)# system-diagnostics event-trace  enable
```

Note
If you disable the *event_trace_state*, you save approximately 1 KB of database storage for each SMF database record.

Step 2 [*Optional*] Use the **show running-config system-diagnostics event-trace** command to verify whether the event trace is enabled.

Example:

```
 [smf] smf# show running-config system-diagnostics event-trace
 system-diagnostics event-trace enabled
```

Limitations of CDL configuration

This section describes the limitations of CDL configuration.

Limitations on retry mechanism

SMF stops retrying the request in these scenarios:

- By default, the maximum retry limit for a request is three. After reaching this limit, SMF stops retrying the request, and the response is sent to the application
- The system stops retries upon receiving a successful response during the retry, and sends the response to the application.

Limitations on failure handling

SMF performs the failure handling with these limitations:

- The SMF service reroutes calls only when it encounters gRPC errors, such as UNAVAILABLE. It does not acknowledge errors returned by the datastore endpoint or actual gRPC timeouts, such as the DEADLINE_EXCEEDED gRPC status code.
- The SMF service does not resolve failures that occur with the datastore such as indexing and slot failures. The CDL layer must resolve these failures, and send a remote API call if necessary.