

Subscriber migration from CPS 7.5 to cnAAA

- Subscriber migration from CPS 7.5 to cnAAA (Solution 1), on page 1
- Subscriber migration from CPS 7.5 to cnAAA (Solution 2) (SOAP Kafka Relay), on page 7

Subscriber migration from CPS 7.5 to cnAAA (Solution 1)

Feature History

Feature Name	Release Information	Description
Subscriber migration from CPS 7.5 to cnAAA	2025.04.0	This feature migrates subscriber management and policy enforcement from CPS 7.5 to the cnAAA platform to ensure service continuity during the transition. The migration is performed in phased, circle-by-circle upgrades with a SOAP proxy managing traffic and enabling temporary shared SPR access.

This feature migrates subscriber management and policy enforcement from CPS 7.5 to the cnAAA platform. It ensures service continuity during migration, intelligent API routing between platforms, and reliable rollback mechanism. It addresses key migration challenges, including staggered BNG transitions, single IP address constraints, and session data inconsistencies between platforms. A SOAP proxy manages traffic and allows temporary shared SPR access. This phased, circle-by-circle upgrade minimizes service disruption.

How subscriber migration works

This process describes how the system migrates subscriber data and sessions from CPS 7.5 to cnAAA.

1. **Initial traffic flow**: OCS and EMA send traffic directly to CPS 7.5 which manages subscriber data and sessions.
2. **Pre migration setup**:
    - The cnAAA component temporarily uses the SPR from Cisco Policy Suite 7.5.

- Reconfigure the OCS and EMA to point to the new SOAP Proxy, instead of directly to Cisco Policy Suite 7.5.

3. **API request handling by SOAP proxy during transition**:

   SOAP proxy intercepts all subscriber API requests (Create, Get, Update, Delete) from OCS and EMA and forwards them to CPS 7.5.

   - CPS 7.5 processes the **Create** or **Get** requests and sends a response to SOAP proxy which is then forwarded to both the OCS and the EMA. The cnAAA component is not involved in this process.
   - For **Update** and **Delete** requests, the SOAP proxy forwards them to CPS 7.5. If CPS 7.5 processes the request successfully, the SOAP proxy extracts the subscriber's MAC address and takes action:
     - For **Update** requests, it triggers a **Refresh Profile API** to cnAAA, which then processes any necessary CoA using the shared CPS 7.5 SPR and its CDL.
     - For **Delete** requests, it triggers a **Delete Session API** to cnAAA, which then removes the corresponding session from its CDL.

4. **Staggered BNG or circle migration**:

   The system gradually reconfigures BNGs within a circle to point to cnAAA. During this phase, some subscribers remain on CPS 7.5, while others move to cnAAA. The SOAP proxy routes requests for all subscribers.

5. **Circle completion**:

   Once all BNGs in a circle migrate to cnAAA, the OCS and EMA for that circle bypass the SOAP proxy and point directly to cnAAA. This completes the circle's transition.

6. **Overall Migration Completion**:

   After all circles migrate and point directly to cnAAA, the system exports the CPS 7.5 SPR database and imports it into cnAAA's local SPR. The system can then decommission CPS 7.5 for subscriber management.

HQoS Turbo Plan Rollout for Subscribers

This feature automates the migration of existing subscribers to new, higher-quality (HQoS) "Turbo Plans." It simplifies the process of upgrading customer services by enables internet service providers to apply enhanced bandwidth and QoS profiles across the subscriber base. This helps large-scale service upgrades and enhances the experience by minimizing manual effort.

Prerequisites:

Before executing the migration script, ensure the following prerequisites are met:

- **Policy Builder Configuration**: Add a new "Service Option" for the "Turbo Plan" to each HSI (High-Speed Internet) Service in Policy Builder for all relevant zones.
- **Execution Environment**: Run the HQoS_Migration_Script.py script from the master node of a setup with Persistent Volume (PV) enabled. Use master node-1 to execute the script.

Script execution

Procedure

To run the HQoS Migration Script, follow these steps:

Step 1 Create an input file named migrationData.csv which must contain comma-separated values with these columns:

- BNG_IP
- Service_Name

Example:

```
BNG_IP,Service_Name
192.168.101.98,TurboPlan_HSI_Service1
192.168.101.99,TurboPlan_HSI_Service2
```

Step 2 Go to the directory where the HQoS_Migration_Script.py file is located. and provide the full path to the script in the command.

Step 3 Run this command syntax to execute the script:

```
/data/pcf-m13/data-pcf-utilities-0/support/script/python
HQoS_Migration_Script.py --migrationfile=<path_to_migrationData.csv>
 --tps=<transactions_per_second> --port=<port_number> --soapurl=<unified_api_soap_url>
--opscentreip=<opscentre_ip_address> --pcfNamespace=<pcf_namespace>
```

How to verify PV enablement on the master node

Procedure

To verify Persistent Volume (PV) enablement on your Kubernetes master node, follow these steps:

Step 1 Access the Master Node and Verify `kubectl` Configuration.

Step 2 Enter the `kubectl get pv` command to list all PV configured in the cluster.

Note The presence of PVs in the output indicates that they are enabled. A `Bound` status confirms active allocation.

Step 3 Run the `kubectl get pvc -A` command to list all PV Claims across all namespaces.

Note PV Claims in a `Bound` state show that applications are successfully using storage. This means the PV is not only available but is also being used by workloads.

Step 4 Enter the `kubectl get sc` to verify the storage classes. **(Optional)**

Configure subscriber migration

Configure the SOAP Proxy, define API URLs, and establish database connectivity for the migration process.

Follow these steps to configure CPC migration:

Procedure

Step 1 Login to the Ops-Center and enter configuration mode.

```
config
```

Step 2 Enable the CPC migration and view available options.

a) **Enable migration**: `cpc-migration enable true`

b) **View options**: Use the `show full-configuration cpc-migration` command to display all available options and their descriptions.

```
cpc-migration enable [true | false]
```

Step 3 Configure the SOAP Proxy network access.

a) **Set external IP addresses**: `cpc-migration external-ips <IP1> <IP2>`

b) **Set the server port**: `cpc-migration properties server.port value <port value>`

Note Once configured, the SOAP Proxy can be accessed using the URL: `http://<configured_external_IP>:<configured_Server_Port>/soap`.

Step 4 Set the Unified API URLs for CPS 7.5 and cnAAA.

a) **CPS 7.5 URL**: `cpc-migration properties cps.url value <SOAP_URL_OF_CPS_7.5>/soap`

b) **cnAAA URL**: `cpc-migration properties cnaaa.url value <SOAP_URL_OF_cnAAA>/soap`

Example:

```
cpc-migration properties cps.url value https://192.0.2.184:8443/ua/soap
cpc-migration properties cnaaa.url value http://pcf-unified-api:8080/ua/soap/soap
```

Step 5 **Define SPR MongoDB connectivity**: Use the `cpc-migration ip-hostnames` command to associate IP addresses with their respective hostnames.

a) **Primary node**: `cpc-migration ip-hostnames <primary IP> hostname sessionmgr01-HA6`

b) **Secondary node**: `cpc-migration ip-hostnames <secondary IP> hostname sessionmgr02-HA6`

Step 6 Configure subscriber and replica properties.

a) **Enable GetSubscriber:** `cpc-migration properties enable.getsubscriber value true`

b) **Set replicas:** `cpc-migration replicas 3` (You can set a value from 1 to 5).

Step 7 (Optional) Configure debug logging.

a) **Set the logger port:** `cpc-migration logger-port <port number>`

b) **Set logging levels:**

```
cpc-migration debug logging logger com.cisco.cpc.migration.proxy
  level debug
exit
```

Step 8 Commit the changes and exit configuration mode.

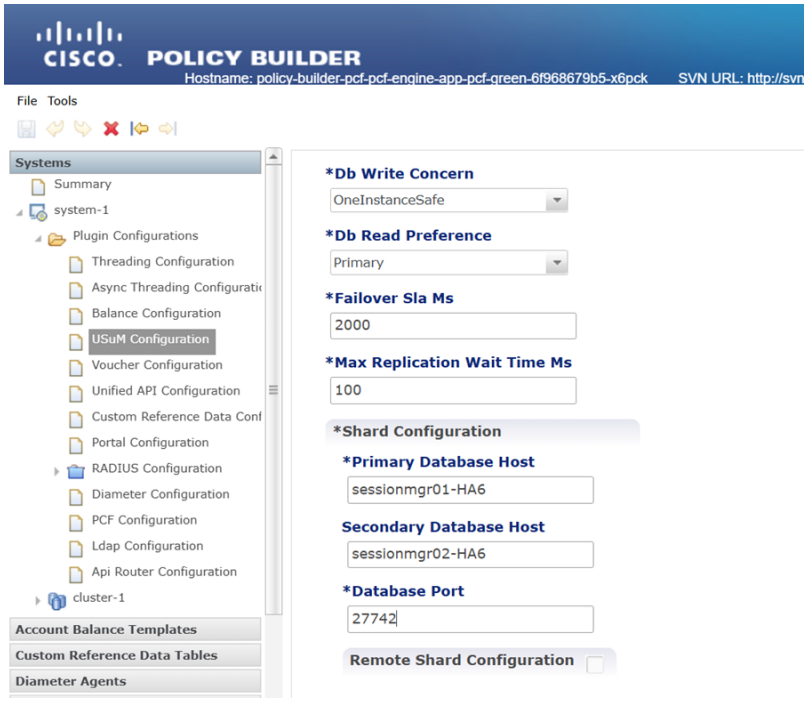
```
commit
exit
```

Note Once the system is up, configure the USum settings to connect to the CPS 7.5 SPR. This involves updating shared configurations with the primary database host, secondary database host, and database port of the CPS 7.5 SPR.

Note After configuring, delete the engine pods and control center pods this will restart them with the new configurations.

Policy Builder Configuration for USum

To connect cnAAA to the existing CPS 7.5 SPR database, the Policy Builder's USum settings must be configured.



Procedure

Step 1 In the Policy Builder GUI, navigate to **Plugin Configuration** > **USum configurations.**

Step 2 Enter the **Primary Database Host** configured in Ops Center.

Step 3 Enter the **Secondary Database Host** configured in Ops Center.

Step 4 Enter the **Database Port**.



Note To view CPS 7.5 SPR data in the cnAAA Control Center, remove the `-Dcc.ua.soap.url=<nil>` property from the Ops-Center configuration if it is present.

Logs available for migration

When subscriber migration is enabled from CPS 7.5 to the cnAAA, a dedicated `api-proxy-logging-0` Kubernetes pod deploys to capture and preserve logs from the SOAP API proxy pod. This ensures that log data, essential for debugging, monitoring, and tracking subscriber migration progress, persists even after SOAP API Proxy pod restarts.

These are the available log files for monitoring subscriber migration:

- **cnaaa-unfiedapi.log**: This file contains informational logs about requests sent to and responses received from the cnAAA Unified API, including Refresh Profile and Delete Session API calls.

  File path: `/data/api-proxy/cnaaa-unfiedapi.log`

  ```
  INFO log.cnAAA.unifiedAPI - Sending refresh_profile request <?xml version="1.0"
  encoding="UTF-8"?>
  <se:Envelope
  xmlns:se="http://schemas.xmlsoap.org/soap/envelope/"><se:Body><RefreshSubscriberProfileRequest


  xmlns="http://broadhop.com/unifiedapi/soap/types"><networkId><![CDATA[0005.9A3C.7A00]]></networkId></RefreshSubscriberProfileRequest>
  </se:Body></se:Envelope> , from source IP Address: 192.168.64.64
  ```

- **cps-unfiedapi.log**: This file captures informational logs about requests sent to and responses received from the CPS 7.5 Unified API. It details the primary processing of subscriber operations (Create, Get, Update, Delete) by CPS 7.5.

  File path:`/data/api-proxy/cps-unfiedapi.log`

  ```
  INFO  log.cps.unifiedAPI - Received response: <se:Envelope
  xmlns:se="http://schemas.xmlsoap.org/soap/envelope/">
  <se:Body><CreateSubscriberResponse xmlns="http://broadhop.com/unifiedapi/soap/types">
  <errorCode>0</errorCode><errorMessage>Request completed
  successfully</errorMessage></CreateSubscriberResponse></se:Body>
  </se:Envelope> for Source IPAddress: 10.1.41.37
  ```

- **application.log**: This is a comprehensive application log file. It includes general operational information and specific details about the SOAP Proxy's processing logic and its interactions with CPS and cnAAA.

  File path: `/data/api-proxy/application.log`

```
INFO  c.c.cpc.migration.proxy.handler.SoapProxyHandler - Processing UPDATE_SUBSCRIBER
operation and status of
EnableGetSubscriber false
```

:

Subscriber migration from CPS 7.5 to cnAAA (Solution 2) (SOAP Kafka Relay)

Feature History

Feature Name	Release Information	Description
Subscriber migration from CPS 7.5 to cnAAA (SOAP Kafka Relay)	2026.01.0	The SOAP Kafka Relay solution for subscriber migration from CPS 7.5 to cnAAA, is an advanced solution which addresses the limitations of subscriber migration solution that uses the common CPS 7.5 SPR until the migration is completed. It addresses • performance and stability issues from shared database usage in CPS 7.5 and cnAAA, • introduces an asynchronous, Kafka-based replication mechanism to decouple database operations, and • ensures data consistency, system stability, and reduced operational risk during subscriber migration.

The SOAP Kafka Relay migrates subscriber data by functioning as a dual-write proxy. It synchronously services the primary legacy system (CPS 7.5) and asynchronously replicates changes to the new system (cnAAA). This decoupling prevents performance issues associated with shared database access.

Operational logic and monitoring

Core components

The architecture includes three primary components deployed within a Kubernetes environment:

- **SOAP proxy service (producer)**: Acts as the entry point for all Online Charging System (OCS) or Element Management System (EMA) SOAP traffic. It performs a synchronous write to CPS 7.5. If CPS

7.5 returns a success code (`errorCode 0`), the proxy publishes the request to the Kafka cluster and returns a success response to the client.

- **Kafka cluster:** Functions as a high-performance, durable buffer. It uses a partitioning strategy based on `networkId` to ensure that the system processes all messages for a specific subscriber in the exact order they were received. It also maintains a Dead Letter Queue (DLQ) for messages that fail replication after multiple attempts.
- **Consumer service:** Provides an asynchronous service that reads messages from the Kafka cluster. It executes the business logic required to update cnAAA and manages retries for downstream timeouts.

Data consistency

To ensure that cnAAA remains synchronized with CPS 7.5, the consumer service uses this process for updates:

- The service receives an `UpdateSubscriber` request.
- The service sends a `GetSubscriber` call to cnAAA.
- The service retrieves the current `<id>` and `<version>` from cnAAA.
- The service injects these values into the original update request and sends the request to cnAAA. This prevents version mismatches and ensures data integrity during the migration period.

High availability and Geo-Redundancy

The solution provides multi-site resilience using an active-passive model:

- MirrorMaker 2 (MM2): Replicates Kafka topics and consumer offsets between Site A and Site B.
- Heartbeat Mechanism: The passive site monitors heartbeats from the active site. If the active site becomes unavailable, the passive site detects the loss of heartbeats and activates its mirror or remote consumer group.
- Deterministic fail-over: Because offsets are replicated, the consumer at the secondary site identifies exactly where the primary site stopped. This ensures at-least-once delivery without data loss.
- Fail-back logic: When the primary site recovers, it compares local offsets with the processed offsets of the remote site to ensure it resumes from the most recent transaction.

Configure subscriber migration parameters in Ops-Center

Configure the SOAP Kafka Relay solution by performing this procedure. The parameters control dual-write behavior, Kafka messaging, rate limiting, and Geo-Redundancy settings.

Before you begin

Disable the previous migration solution before enabling the SOAP Kafka Relay.

```
set cpc-migration enable false
```

Procedure

Step 1 Log in to Ops-Center and enter the configuration mode.

```
config
```

Step 2 Configure core migration and server parameters.

a. Enable the SOAP Kafka Relay solution.

```
set cpc-migration soap-kafka-proxy enable true
```

b. Configure the SOAP proxy port and IP.

```
set cpc-migration soap-kafka-proxy port 8080
set cpc-migration soap-kafka-proxy external-ips [ <IP_Address> ]
```

Step 3 Configure pod affinity labels.

- `set cpc-migration soap-kafka-proxy affinity-label key <key_name>`

  `set cpc-migration soap-kafka-proxy affinity-label value <value_name>`

- Configure the label to the nodes where Kafka pods need to be deployed.

  `kubectl label node <node-name> <key_name>=<value_name>`

Step 4 Define primary and secondary URLs.

- The primary URL points to the active CPS 7.5 instance.
- The secondary URL points to the standby CPS 7.5 instance.

```
cpc-migration soap-kafka-proxy producer properties producer.primary.url
 value http://cps-7-5.example.com:8001/ua/soap
exit
cpc-migration soap-kafka-proxy consumer properties consumer.primary.url
 value http://pcf-unified-api:8080/ua/soap
exit
```

- Enable and configure the cnAAA secondary site URL for fail-over.

```
cpc-migration soap-kafka-proxy consumer properties secondary.url.enabled
 value true
exit

cpc-migration soap-kafka-proxy consumer properties consumer.secondary.url
 value http://pcf-unified-api:8080/ua/soap
exit
```

Step 5 Configure Geo-Redundancy

a. Enable cluster coordination for the consumer.

```
cpc-migration soap-kafka-proxy consumer properties cluster.coordination.enabled value <true|false>
```

b. Specify if the current site is the primary site.

```
cpc-migration soap-kafka-proxy consumer properties is.primary.site value <true|false>
```

c. Configure the local and remote Kafka site identifiers.

```
cpc-migration soap-kafka-proxy kafka local-site-id <Local_Site_Name>
cpc-migration soap-kafka-proxy kafka remote-site-id <Remote_Site_Name>
```

d. Configure the local Kafka external IP addresses.

```
cpc-migration soap-kafka-proxy kafka external-ip <local_Kafka_IP> <port>
exit
```

e. Configure the remote Kafka server IP addresses.

```
cpc-migration soap-kafka-proxy kafka remote-site kafka-server <remote_Kafka_IP> <port>
exit
```

Note Do not change the configured site-ID names. Changing these names creates separate Kafka topics for each ID, which impacts replication.

Step 6 Review the configuration to confirm that all URLs and site IDs are correct. After verifying, `commit` the changes.

```
show full-configuration cpc-migration soap-kafka-proxy
commit
```

Alerts

The system provides these alerts for monitoring and troubleshooting:

Critical Alerts

- Kafka consumer lag is consistently high or growing for more than 5 minutes.
- High rate of failures when sending requests to the primary or secondary systems.

Warning Alerts

High rate of messages sent to the DLQ.

DLQ processor script

The `subscriber_migration_dlq_processor.py` script retrieves subscriber messages from the `FailedApi-DLQ` Kafka topic. It then replays the contained SOAP requests against a specified consumer SOAP endpoint. Use this utility to recover or replay failed subscriber migration SOAP requests that are captured in the Dead Letter Queue (DLQ). This is helpful during troubleshooting or after you apply fixes to the consumer or service.

The script is located in the `/data/utilities/support/script/` directory within the `pcf-utilities` pod.

Run the DLQ processor

- Log in to the `pcf-utilities-0` pod from your master node:

  ```
  kubectl exec -it -n <namespace> pcf-utilities-0 -- bash
  ```

- Navigate to the script directory.

  ```
  cd /data/utilities/support/script/
  ```

- Run the script without arguments and follow the on-screen prompts to specify the namespace, input type, value, and consumer URL.

  ```
  python3 subscriber_migration_dlq_processor.py
  ```

- Run the script with specific arguments.

Table 1: Command-line examples

Processing option	Command
Fetch the last 100 messages	`python3 subscriber_migration_dlq_processor.py 1 100 "https://unifiedapi/ua/soap" <namespace>`
Process from a specific Message ID	`python3 subscriber_migration_dlq_processor.py 2 <message-id> "https://unifiedapi/ua/soap" <namespace> --no-verify-ssl`
Process from a specific timestamp	`python3 subscriber_migration_dlq_processor.py 3 <timestamp> "https://unifiedapi/ua/soap" <namespace> --no-verify-ssl`