

HTTP API Reference

The OmniRoam API serves TAP files, RAP files, CDR views, totals, export control, and metrics. This guide lists the endpoints.

Table of Contents

- [Access](#)
- [TAP Files](#)
- [TAP Validation](#)
- [RAP Files](#)
- [CDR Views](#)
- [CDR Upload](#)
- [Totals and Monitoring](#)
- [TAP Export](#)
- [Metrics](#)

Access

The OmniRoam API listens on port 4000. The API serves plain HTTP, without TLS. All endpoints are under the `/api` path. The API returns JSON without an envelope.

Resource	Path
Swagger user interface	GET <code>/api/docs</code>
OpenAPI schema	GET <code>/api/schema</code>

TAP Files

Method and path	Purpose	Parameters
GET /api/tap_files_in	List the inbound TAP files.	tadig (optional)
GET /api/tap_files_in/:filename	Return one inbound TAP file as JSON.	tadig, page, per_page
GET /api/tap_files_out	List the outbound TAP files.	tadig (optional)
GET /api/tap_files_out/:filename	Return one outbound TAP file as JSON.	tadig, page, per_page

The file response holds the header and one page of call events. The `page` value starts at `1`. The `per_page` value has a default of `100` and a maximum of `2000`. The response returns `404` when the file is not found.

TAP Validation

Method and path	Purpose	Parameters
GET /api/tap_validate_in/:filename	Validate an inbound TAP file.	tadig (optional)
GET /api/tap_validate_out/:filename	Validate an outbound TAP file.	tadig (optional)

The response reports whether the file is valid, the count of findings by severity, and the list of findings. See [TAP Validation](#).

RAP Files

Method and path	Purpose	Parameters or body
GET /api/rap_files	List the RAP files.	tadig (optional)
GET /api/rap_files/:filename	Return one RAP file as enriched JSON.	tadig (optional)
POST /api/rap_files/send_alert	Send an email alert for a RAP file.	Body: filename (required), tadig

See [RAP Processing](#).

CDR Views

Method and path	Purpose	Parameters
GET /api/part_cdrs/:page	List the in-flight partial CDR keys from Redis.	Path: page (integer)
GET /api/cdrs_from_csv	List the raw CDR files, or download one.	filename, raw_cdr_view_max_age
GET /api/config_yaml	Return config.yaml. The API removes the InfluxDB token.	None
GET /api/config	Return the config section and the counters.	None

CDR Upload

Method and path	Purpose	Body
POST <code>/api/ui_upload</code>	Upload a CSV CDR file to the OmniCore directory.	Multipart <code>file</code> . The name must be a bare filename ending <code>.csv</code> — no path separators. Returns 400 (rejected), 409 (already held), 413 (too large).

The response returns `200` on success, `409` for a duplicate file, and `500` for an invalid file. See [CDR Processing](#).

Totals and Monitoring

Method and path	Purpose	Parameters
GET <code>/api/monitoring</code>	Return monitoring totals, or download a matching file.	<code>filename</code> (optional)
GET <code>/api/totals_monthly</code>	List the partners with monthly totals.	None
GET <code>/api/totals_monthly/:tadig</code>	Return the monthly totals for a TADIG.	<code>year</code> , <code>month</code>
GET <code>/api/totals_daily</code>	Return the daily totals across all TADIGs.	None
GET <code>/api/totals_daily/:tadig</code>	Return the daily totals for a TADIG.	None

TAP Export

Method and path	Purpose	Body
POST /api/tap_export	Queue an export for one partner.	partner (required), records_per_file, max_files
GET /api/tap_export	List the export jobs.	None
GET /api/tap_export/:job_id	Return one export job.	None
GET /api/export_schedule	Return the scheduler status and the queue state.	None
POST /api/export_schedule	Queue an export for every partner now.	None

The export endpoints return `202` when OmniRoam queues the job. The job queue runs one export at a time. See the [Operations Guide](#).

Metrics

Method and path	Purpose
GET /api/metrics	Return the Prometheus metrics.

The response is Prometheus text exposition. A scrape reads a cached snapshot. See the [Metrics Reference](#).

- Overview

Architecture

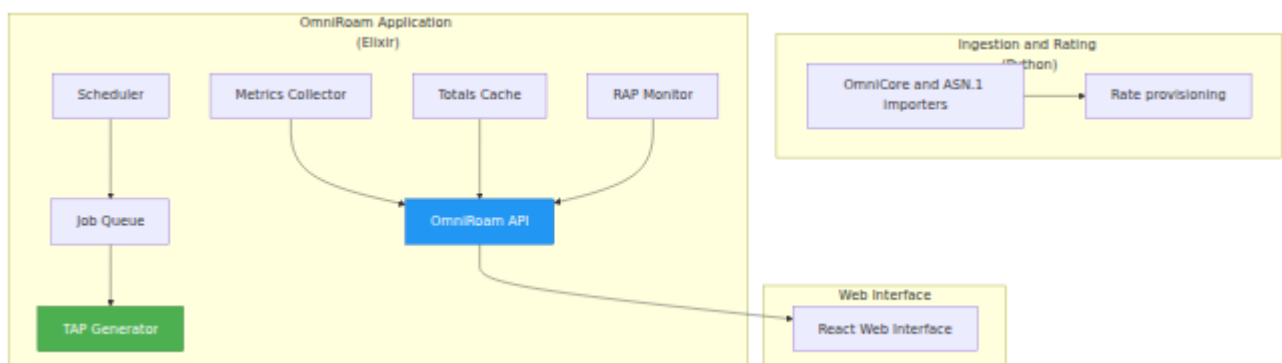
This guide describes the OmniRoam components, the end-to-end data flow, and the external systems that OmniRoam integrates with.

Table of Contents

- [Component Overview](#)
- [Deployable Packages](#)
- [Data Flow](#)
 - [Ingestion and Rating](#)
 - [TAP File Generation](#)
 - [File Transfer](#)
 - [RAP Return](#)
- [Integration Points](#)

Component Overview

OmniRoam has two parts. A set of Python tools ingests and rates CDRs. An Elixir application generates TAP files, processes RAP files, and serves the API and metrics.



Ingestion and rating (Python)

These tools run as scheduled jobs. They do not run as long-lived services.

- **OmniCore importer** - Reads OmniCore serving gateway CSV files, correlates the partial records into complete data sessions, and writes the sessions to Redis.
- **ASN.1 importer** - Converts 3GPP ASN.1 CDR files to XML and imports them.
- **Redis importer** - Drains the Redis queue, applies age filters, rounds the usage, and sends each record to the Charging Engine for rating.
- **Rate provisioner** - Loads the per-partner rate plans into the Charging Engine from `config.yaml`.

OmniRoam application (Elixir)

The OmniRoam application runs as a systemd service. It starts a supervised set of processes.

Process	Role
File Index	Caches parsed TAP and RAP file metadata for fast listing.
Totals Cache	Caches per-partner monthly and daily totals.
Metrics Collector	Builds the Prometheus metrics snapshot in the background.
Job Queue	Runs one TAP export at a time.
Scheduler	Starts the daily export at the configured time.
OmniRoam API	Serves the HTTP API on port 4000.
RAP Monitor	Scans for new RAP files and sends alerts.
Sequence	Reserves TAP file sequence numbers and writes <code>counters.yaml</code> .
Space Stripper	Cleans stray spaces from the rated CDR fields in the database.

The **TAP Generator** reads rated CDRs, maps each CDR to a TAP record, assembles the batch, encodes the batch as a binary TAP file, and marks the CDRs as processed. See [TAP File Generation](#) for details.

Web interface

The web interface is a React single-page application. It is served as static content by nginx. The web interface calls the OmniRoam API on port 4000.

Registrar

The registrar is an optional outbound connector. The registrar opens a WebSocket to a management system and forwards that system's requests to

the local OmniRoam API. Use the registrar to reach the API from behind NAT without an inbound port.

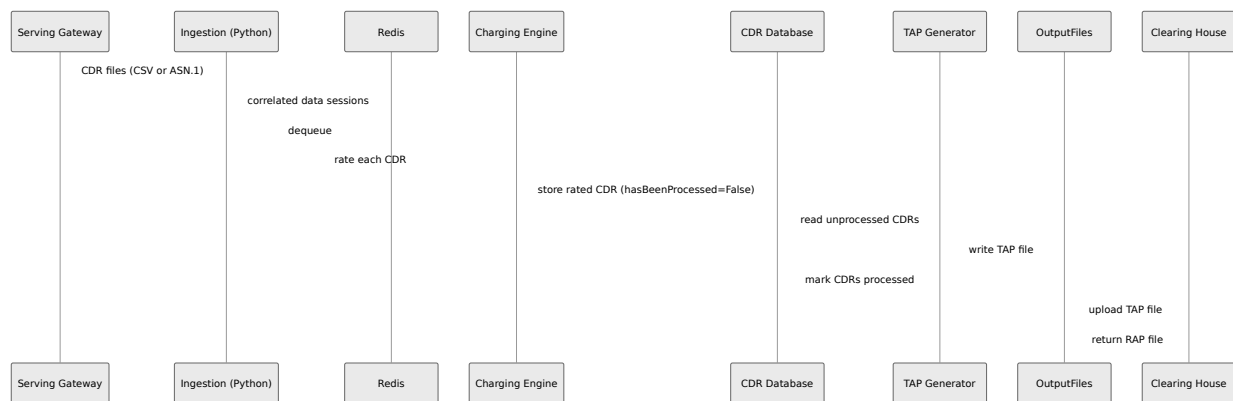
Deployable Packages

OmniRoam ships as a set of Debian packages. The `omniroam` meta-package depends on the component packages and locks their versions together.

Package	Contents	Runtime
<code>omniroam</code>	Meta-package. Depends on the component packages.	None.
<code>omniroam-api</code>	The Elixir application release.	systemd service on port 4000, user <code>omniroam</code> .
<code>omniroam-generator</code>	The Python ingestion and rating tools plus configuration.	No service. Run by cron or by hand.
<code>omniroam-ui</code>	The static web interface and the nginx site.	nginx on port 80.
<code>omniroam-registrar</code>	The outbound registrar connector.	systemd service.

Data Flow

A CDR travels through four stages: ingestion and rating, TAP file generation, file transfer, and RAP return.



Ingestion and rating

Two CDR sources feed one pipeline.

1. **OmniCore serving gateways** send CSV CDR files. The files arrive by rsync or by an HTTP upload to the API. OmniRoam writes the files to a local directory.
2. **3GPP ASN.1 CDR sources** provide ASN.1 CDR files. OmniRoam pulls the files and converts them to XML.
3. The importer correlates the partial records, writes complete data sessions to Redis, and then drains Redis into the Charging Engine for rating.

Rated CDRs are stored in the Charging Engine database (the CDR table). Each CDR carries a `hasBeenProcessed` marker. A value of `False` means the CDR still needs a TAP file. See [CDR Processing](#) for the ingestion and rating detail.

TAP file generation

The scheduler or an API request enqueues an export job. The job queue runs one export at a time. The generator reads the unprocessed CDRs for the partner, oldest first, and writes one or more TAP files. The generator then marks the CDRs as processed and records the file sequence number. See [TAP File Generation](#).

File transfer

When the job queue drains, OmniRoam runs the upload command. The upload command sends the new TAP files to the clearing house. A separate pull job

retrieves inbound TAP and RAP files. See the [Operations Guide](#) for the transfer jobs.

RAP return

The partner validates each TAP file. If the partner rejects records, the partner returns a RAP file. The pull job places the RAP file in the inbound directory. The RAP monitor scans the directory, reads each new RAP file, and sends an email alert. See [RAP Processing](#).

Integration Points

System	Protocol	Purpose
Redis	Redis on localhost	Queue of correlated, partial CDRs before rating.
Charging Engine	JSON-RPC on port 2080	Rates CDRs and returns the cost.
CDR database	Database on port 3306	Stores rated CDRs and marks them processed.
InfluxDB	HTTP API	Stores usage and export analytics.
Clearing house	SCP	Sends TAP files and pulls inbound TAP and RAP files.
Email provider	HTTPS	Sends RAP alert email.
Prometheus	HTTP scrape on <code>/api/metrics</code>	Collects OmniRoam metrics.

See the [HTTP API Reference](#) for the endpoints, and the [Configuration Reference](#) for the connection settings.

- [Overview](#)

CDR Processing

This guide describes how OmniRoam ingests Call Detail Records (CDRs), rates them, and stores them for TAP file generation.

Table of Contents

- [CDR Sources](#)
- [Ingestion Pipeline](#)
- [Age Filters](#)
- [Usage Rounding](#)
- [Rating](#)
- [Storage](#)

CDR Sources

OmniRoam ingests CDRs from two source types.

Source	Format	Transfer	Landing directory
OmniCore serving gateway	CSV	rsync, or an HTTP upload to the API	<code>/var/OmniCore_CDRs</code>
3GPP ASN.1 CDR source	ASN.1	rclone	<code>/var/asn1_cdrs</code>

A serving gateway can also send a CSV file directly to the API with a `POST /api/ui_upload` request. OmniRoam saves the uploaded file to the OmniCore directory. See the [HTTP API Reference](#).

Ingestion Pipeline

The ingestion tools run as scheduled jobs. See the [Operations Guide](#) for the schedule.



1. **Correlate.** A single data session can produce more than one partial CDR. The importer joins the partial records for one session into one complete record.
2. **Queue.** The importer writes each complete record to Redis.
3. **Filter.** The Redis importer drains the queue and removes records that fall outside the age window. The importer also removes records that have zero usage.
4. **Round.** The importer rounds the usage. See [Usage Rounding](#).
5. **Rate.** The importer sends each record to the Charging Engine for pricing. See [Rating](#).
6. **Store.** The Charging Engine stores the rated CDR in the CDR table. The importer also writes a usage point to InfluxDB.

Age Filters

The Redis importer applies two age filters before it rates a record.

Filter	Action
Older than 30 days	The importer removes the record. A TAP file cannot include a CDR that is more than 30 days old.
Younger than 24 hours	The importer defers the record. The importer processes the record on a later run.

The 30-day limit comes from the GSMA rules for TAP files. Ingest and rate CDRs promptly, or the 30-day limit will exclude them from settlement.

Usage Rounding

OmniRoam rounds the usage of each CDR up to the next multiple of `round_up_to` bytes. The `round_up_to` value is set per partner. See the [Configuration Reference](#).

OmniRoam applies two separate roundings.

1. **Per-direction rounding.** OmniRoam rounds the incoming volume and the outgoing volume up to the next multiple, each on its own.
2. **Charged units.** OmniRoam sums the original incoming and outgoing volumes, and then rounds that total up to the next multiple. This rounded total is the charged usage that the Charging Engine prices.

OmniRoam keeps the original, unrounded volumes for audit.

Example. A CDR has 500 bytes incoming and 700 bytes outgoing. The partner uses `round_up_to: 1024`.

- Per-direction: incoming rounds to 1024, outgoing rounds to 1024.
- Charged units: the total is 1200 bytes, which rounds to 2048 bytes.

Rating

OmniRoam prices each CDR with the Charging Engine. Rating happens during ingestion, not during TAP file generation.

1. The rate provisioner loads each partner's rate into the Charging Engine. The rate comes from `unit_price` and `unit_bytes` in the partner configuration.
2. The importer sends each CDR to the Charging Engine with the charged usage.
3. The Charging Engine returns the cost. OmniRoam stores the cost with the CDR.

To change a partner's price, update `unit_price` or `unit_bytes` in the [Configuration Reference](#), and then reload the rates into the Charging Engine.

Storage

OmniRoam stores a rated CDR in the Charging Engine database. Each CDR carries a `hasBeenProcessed` marker.

Marker value	Meaning
<code>False</code>	The CDR is rated but not yet in a TAP file.
A file name	The CDR is in the named TAP file.

The generator reads the CDRs with `hasBeenProcessed` set to `False`, oldest first. See [TAP File Generation](#).

OmniRoam also writes usage analytics to InfluxDB. See the [Metrics Reference](#).

-
- [Overview](#)

Configuration Reference

This guide documents every parameter in the OmniRoam configuration files. OmniRoam reads two files:

- `config.yaml` - The main configuration. It has a global `config` section and a per-partner `partners` section.
- `counters.yaml` - The TAP file sequence counters.

The Python tools read `config.yaml` from the working directory. The Elixir application reads `config.yaml` from the path in the `OMNIROAM_CONFIG_PATH` environment variable, or from `/etc/pytap3/config.yaml`. OmniRoam reads `counters.yaml` from the same directory as `config.yaml`.

Table of Contents

- [Configuration Structure](#)
- [Global Configuration](#)
 - [TAC Configuration](#)
 - [Home PLMNs](#)
 - [Processor Identity and Timezone](#)
 - [File Paths](#)
 - [Raw CDR View](#)
 - [InfluxDB Connection](#)
 - [Logging](#)
 - [Export Schedule](#)
 - [Export Completion Command](#)
 - [Webhooks](#)
 - [Records Per File](#)
- [Partner Configuration](#)
 - [IMSI Prefixes](#)

- Access Point Name
- Rates
- Usage Rounding
- Batch Information
- Accounting Information
- Call Type Levels
- Sequence Counters
- Deprecated Keys

Configuration Structure

```
config:  
  # Global system configuration  
  
partners:  
  # Per-partner configuration
```

Global Configuration

TAC Configuration

`config.tac_config` maps a Tracking Area Code (TAC) to a serving location. OmniRoam uses the serving location to set the billing identifier and the timezone in each TAP record. Each entry is a named template with a list of TACs.

```
config:  
  tac_config:  
    ExampleNetwork:  
      tac_list: ['1101', '10000', '10100']  
      servingBid: 12345  
      servingLocationDescription: 'Example Network'  
      timezone: 'Etc/UTC'
```

Parameter	Type	Required	Default	Description
<code>tac_list</code>	List of String	Yes	-	The TAC values that match this template. OmniRoam matches a CDR's TAC against the list.
<code>servingBid</code>	Integer	No	-	The 5-digit billing identifier written to the TAP record. Om to leave the field out.
<code>servingLocationDescription</code>	String	No	-	The location description written to the TAP record.
<code>timezone</code>	String	Yes	-	The IANA timezone name. OmniRoam uses this timezone to compute the UTC offset for matching CDRs.

Home PLMNs

`config.home_plmns` lists the home PLMN prefixes. OmniRoam treats a CDR whose IMSI starts with a home PLMN prefix as a home subscriber, not a roaming subscriber. The ingestion tools read this key.

```
config:
  home_plmns:
    - '001010'
```

Parameter	Type	Required	Default	Description
<code>home_plmns</code>	List of String	Yes	-	The IMSI prefixes (the first 6 digits) that identify a home subscriber.

Processor Identity and Timezone

```
config:
  cdr_processor_tz: America/New_York
```

Parameter	Type	Required	Default	Description
<code>cdr_processor_tz</code>	String	Yes	-	The default IANA timezone. OmniRoam uses this timezone for file timestamps and for a CDR that matches no TAC template.

File Paths

```
config:
  cdr_search_paths:
    - '/var/asn1_cdrs/'
    - '/var/OmniCore_CDRs/'
  tap_output_path: '/etc/pytap3/OutputFiles'
  tap_human_readable_output_path: '/etc/pytap3/OutputFiles_Human'
  tap_in_path: '/etc/pytap3/TAP_In'
  roaming_in_filename: False
```

Parameter	Type	Required	Default
<code>cdr_search_paths</code>	List of String	Yes	-
<code>tap_output_path</code>	String	Yes	<code>/etc/pytap3/0u</code>
<code>tap_human_readable_output_path</code>	String	Yes	-
<code>tap_in_path</code>	String	No	<code>/etc/pytap3/TA</code>
<code>roaming_in_filename</code>	Boolean	No	<code>False</code>

Parameter	Type	Required	Default

Raw CDR View

```
config:
  raw_cdr_view_max_age: 7
```

Parameter	Type	Required	Default	Description
<code>raw_cdr_view_max_age</code>	Integer	No	-	The maximum age in days for the raw CDRs that the web interface shows. A request can override this value with a query parameter.

InfluxDB Connection

`config.influx_db` sets the InfluxDB connection for usage and export analytics. All four values must be present, or OmniRoam skips the metric write. See the [Metrics Reference](#) for the measurements.

```
config:
  influx_db:
    influxDbUrl: 'http://influxdb.example.com:8086'
    influxDbOrg: 'example-org'
    influxDbBucket: 'tap3'
    influxDbToken: '<influx-token>'
```

Parameter	Type	Required	Default	Description
<code>influxDbUrl</code>	String	Yes	-	The InfluxDB base URL.
<code>influxDbOrg</code>	String	Yes	-	The InfluxDB organization.
<code>influxDbBucket</code>	String	Yes	-	The InfluxDB bucket.
<code>influxDbToken</code>	String	Yes	-	The InfluxDB API token. The API removes this value from the <code>config_yaml</code> response.

Set `influxDbToken` to your own token. Do not commit a real token to version control.

Logging

```
config:
  log_level: DEBUG
```

Parameter	Type	Required	Default	Description
<code>log_level</code>	String	Yes	-	The log level for the Python tools. Use one of <code>DEBUG</code> , <code>INFO</code> , <code>WARNING</code> , or <code>ERROR</code> .

Export Schedule

`config.export_schedule` controls the daily automatic export. The scheduler reads this section on each tick, so a change applies without a restart. See the [Operations Guide](#) for the scheduler behavior.

```
config:
  export_schedule:
    enabled: false
    at: '02:00'
    partners: all
```

Parameter	Type	Required	Default	Description
<code>enabled</code>	Boolean	No	<code>false</code>	The master switch for the daily export. Set to <code>true</code> to run the export automatically.
<code>at</code>	String	No	-	The daily run time in <code>HH:MM</code> format, in UTC.
<code>partners</code>	String or List	No	<code>all</code>	The partners to export. Use <code>all</code> for every partner, or a list of partner names.

Export Completion Command

`config.on_export_complete` sets a command that OmniRoam runs when the export job queue drains. Use this command to send the new TAP files to the clearing house.

```

config:
  on_export_complete: '/etc/pytap3/smart_tap_upload.sh'

  # Preferred: invoked once per pending TAP file, with the path as
  # the argument
  # and TAP_FILE / TAP_FILENAME / TAP_TADIG / TAP_SENT_DIR in the
  # environment.
  # Exit 0 and OmniRoam moves the file to OutputFiles_Sent; any
  # other status
  # leaves it pending to retry next run. The script does not glob,
  # move files,
  # or track partial failure – so it is usually ten lines. Takes
  # precedence over
  # on_export_complete, which stays for deployments that have not
  # moved over.
  on_tap_file: '/etc/pytap3/send_one.sh'

  # Inbound: run the site's fetch script on a schedule instead of
  # from cron.
  # Invoked with TAP_IN_DIR and RAP_IN_DIR in the environment;
  # exit 0 means the
  # fetch completed (whether or not it found anything). Runs are
  # serialised on
  # their own lock, failures are counted, and
  tap3_inbound_newest_age_seconds
  # makes a clearing house that stopped answering an alert rather
  # than a quiet
  # week. Unset and nothing runs.
  on_fetch_files: '/etc/pytap3/pull_tap.sh'
  fetch_interval_minutes: 60

```

Parameter	Type	Required	Default	Description
on_export_complete	String	No	-	The shell command to run after the export completes. Leave blank to disable.

Webhooks

`config.webhooks` sets Slack incoming webhook URLs. OmniRoam posts a message when an export completes or fails.

```
config:
  webhooks:
    on_complete: '<slack-webhook-url>'
    on_failure: '<slack-webhook-url>'
```

Parameter	Type	Required	Default	Description
<code>on_complete</code>	String	No	-	The Slack webhook URL for a successful export. Leave blank to disable.
<code>on_failure</code>	String	No	-	The Slack webhook URL for a failed export. Leave blank to disable.

Records Per File

`config.records_per_tap_file` sets a global limit on the number of CDRs in one TAP file. A per-partner value and a per-request value override this global value. When a partner has more unprocessed CDRs than the limit, OmniRoam writes more than one TAP file.

```
config:
  records_per_tap_file: 140000
```

Parameter	Type	Required	Default	Description
<code>records_per_tap_file</code>	Integer	No	-	The maximum number of CDRs per TAP file.

Partner Configuration

`partners` holds one entry per roaming partner. The entry name is a label that you choose. The ingestion tools use `imsi_prefixes` to assign a CDR to a partner. The generator uses the partner name to select the partner to export.

```
partners:
  ExamplePartner_Live:
    imsi_prefixes:
      - '001020'
    accessPointName0I: mnc020.mcc001.gprs
    rates:
      unit_price: 0.000476800
      unit_bytes: 1024
    round_up_to: 1024
    batch_info:
      sender: AAAAA
      recipient: BBBBB
      specificationVersionNumber: 3
      releaseVersionNumber: 12
    accountingInfo:
      localCurrency: 'USD'
      tapCurrency: 'USD'
      roundingAction: 'Simple'
      tapDecimalPlaces: 5
    call_type_level:
      qci_1: 21
      default: 20
```

IMSI Prefixes

Parameter	Type	Required	Default	Description
<code>imsi_prefixes</code>	List of String	Yes	-	The IMSI prefixes that map a CDR to this partner. The ingestion tools use the longest match.

Access Point Name

Parameter	Type	Required	Default	Description
<code>accessPointName0I</code>	String	Yes	-	The Access Point Name Operator Identifier. OmniRoam writes this value into each TAP GPRS record.

Rates

`rates` sets the price that OmniRoam loads into the Charging Engine for this partner. The Charging Engine prices each CDR against this rate. See [CDR Processing](#) for the rating flow.

Parameter	Type	Required	Default	Description
<code>unit_price</code>	Float	Yes	-	The price for each <code>unit_bytes</code> of usage, in the local currency.
<code>unit_bytes</code>	Integer	Yes	-	The number of bytes in one rate unit.

Usage Rounding

Parameter	Type	Required	Default	Description
<code>round_up_to</code>	Integer	Yes	-	The rounding unit in bytes. OmniRoam rounds the usage up to the next multiple of this value. See CDR Processing .

Batch Information

`batch_info` sets the identity of each TAP file for this partner.

Parameter	Type	Required	Default	Description
<code>sender</code>	String	Yes	-	Your 5-character code. Omit this value if <code>batchContent</code> and to the
<code>recipient</code>	String	Yes	-	The partner TADIG code. Uses this value to output sub file name, sequence
<code>specificationVersionNumber</code>	Integer	Yes	3	The TAP specification version. Use 3.12.
<code>releaseVersionNumber</code>	Integer	Yes	12	The TAP release. Use 12 for
<code>fileTypeIndicator</code>	String	No	-	When present produces a TAP File Generated this key for

Accounting Information

`accountingInfo` sets the currency and money format of each TAP file.

Parameter	Type	Required	Default	Description
<code>localCurrency</code>	String	Yes	-	The local currency as an ISO 4217 code.
<code>tapCurrency</code>	String	Yes	-	The settlement currency as an ISO 4217 code.
<code>roundingAction</code>	String	Yes	-	The money rounding mode. Use <code>Up</code> , <code>Down</code> , or <code>Simple</code> .
<code>tapDecimalPlaces</code>	Integer	Yes	-	The number of decimal places for the TAP money values. OmniRoam scales charge values by 1000, so set this value to <code>3</code> . See TAP File Generation .

Call Type Levels

`call_type_level` maps a CDR's QoS Class Identifier (QCI) to the TAP call type level. OmniRoam looks up the key `qci_<n>` for the CDR's QCI. If the key is absent, OmniRoam uses `default`.

```
call_type_level:
  qci_1: 21
  qci_2: 22
  default: 20
```

Parameter	Type	Required	Default	Description
<code>qci_1</code> to <code>qci_9</code>	Integer	No	-	The TAP call type level for each QCI value.
<code>default</code>	Integer	Yes	-	The TAP call type level for a QCI that has no <code>qci_<n></code> key.

Sequence Counters

`counters.yaml` holds the TAP file sequence numbers. OmniRoam keys the file by the **recipient TADIG**, with a separate counter for each file type. See [TAP File Generation](#) for the counter behavior.

```
BBBBB:
  CD: 11
  TD: 50
CCCCC:
  CD: 3
  TD: 2
```

Key	Description
Top-level key	The 5-character recipient TADIG code.
<code>CD</code>	The next sequence number for a live (Chargeable Data) file.
<code>TD</code>	The next sequence number for a test (Test Data) file.

Notes:

- Initialize a new partner with `CD: 1` and `TD: 1`.
- OmniRoam increments the counter after it writes a file.

- The sequence number is a 5-digit field. Reset the counter before it reaches `99999`. See the [Operations Guide](#).
- Back up `counters.yaml` with `config.yaml` so that the sequence continues correctly after a restore.

Deprecated Keys

Do not use these keys. OmniRoam no longer reads them in the production path.

Key	Status
<code>config.exporters</code>	Deprecated. The current application does not read this key.
<code>config.cdr_processor_name</code>	Deprecated. The current application does not read this key.

-
- [Overview](#)

Metrics Reference

OmniRoam exposes two metric sources. The OmniRoam API exposes Prometheus metrics for the service state. The ingestion and export tools write usage analytics to InfluxDB.

Table of Contents

- [Prometheus Metrics](#)
- [Service Status](#)
- [Example Queries](#)
- [InfluxDB Measurements](#)

Prometheus Metrics

The OmniRoam API serves Prometheus metrics at `GET /api/metrics`. A scrape reads a cached snapshot, so a scrape does not run any database query. The metrics collector refreshes the snapshot in the background, about once a minute.

All OmniRoam metrics are gauges.

Metric	Labels	Description
tap3_exporter_up	-	The collector has a fresh snapshot. <code>1</code> means yes.
tap3_service_up	service	The service is running. <code>1</code> means up, <code>0</code> means down.
tap3_cgrates_up	-	The Charging Engine API is responding.
tap3_cgrates_goroutines	-	The number of Charging Engine worker routines.
tap3_cdr_total	-	The total number of CDRs in the database.
tap3_files_pending	-	The number of files that wait for upload.
tap3_files_sent	-	The number of files that were sent.
tap3_files_total	-	The total number of TAP files.
tap3_files_generated_today	-	The number of files generated today.
tap3_last_file_generated_timestamp	-	The Unix time of the last file generation.
tap3_last_file_sent_timestamp	-	The Unix time of the last file sent.

Metric	Labels	Description
<code>tap3_partner_files_pending</code>	<code>partner</code>	The number of pending files per partner.
<code>tap3_partner_files_sent</code>	<code>partner</code>	The number of sent files per partner.
<code>tap3_upload_status</code>	-	The last upload status. <code>1</code> is success, <code>0</code> is failure, <code>-1</code> is unknown.
<code>tap3_hours_since_last_file</code>	-	The number of hours since the last TAP file.

The metric `tap3_cdr_total` reads an approximate row count. The collector does not run a full table count, because a full count on the large CDR table starves the export database pool.

Service Status

The `tap3_service_up` metric reports one series for each dependency. The collector checks each service with the system service manager.

Service label	Component
omniroam-api	The OmniRoam API.
cgrates	The Charging Engine (label emitted literally).
mysql	The CDR database (label emitted literally).
redis-server	The ingestion queue.
influxdb	The analytics database.

Example Queries

```
# TAP files that wait for upload
tap3_files_pending

# Hours since the last TAP file, alert when high
tap3_hours_since_last_file > 26

# Partners with pending files
tap3_partner_files_pending > 0

# A dependency that is down
tap3_service_up == 0
```

InfluxDB Measurements

The ingestion and export tools write four measurements to InfluxDB. Set the connection in the [Configuration Reference](#).

Measurement	When it is written	Tags	Fields
raw_cdr	Once per CDR, after rating.	operator, input_file, apn, cellId, imsi, tac, sGWAddress, pGWAddress	chargeableUnit, chargedUnits
omnicore_cdr_part	Once per partial CDR during OmniCore ingestion.	accessPointNameNI, hpmIn, sgw_ip, tac	dataVolumeInco, dataVolumeOutg
file_processing	Once per OmniCore file processed.	individual_filename	no_imsi_count, sgw_cdr_count, pgw_cdr_count, charging_id_mi, roaming_sgw_cd, empty_record, processing_tim
tap_cdr	Once per TAP file exported.	operator, filename	totalcharge, totalconsumed, cdr_count

The `raw_cdr` measurement has high cardinality, because the tags include the IMSI, the cell identifier, and the input file name. Manage the InfluxDB retention policy to control the storage growth. See the [Operations Guide](#).

- [Overview](#)

Operations Guide

This guide describes the OmniRoam scheduled jobs, the file transfer, the archiving, the log locations, and the routine maintenance.

Table of Contents

- [Scheduled Jobs Overview](#)
- [CDR Ingestion](#)
- [Scheduled TAP Export](#)
- [File Transfer](#)
- [Archiving](#)
- [Sequence Counter Maintenance](#)
- [Storage Maintenance](#)
- [Log Locations](#)

Scheduled Jobs Overview

OmniRoam runs on a daily cycle. The table shows the main jobs and their order.

Job	Runs	Purpose
CDR ingestion	Scheduled by cron	Collects, correlates, and rates CDRs.
TAP export	Daily, at the configured time	Generates TAP files per partner.
Upload	After each export	Sends the new TAP files to the clearing house.
Inbound pull	Daily, and a weekly full run	Pulls inbound TAP and RAP files.
RAP monitor	Every 4 hours	Alerts operators to a new RAP file.
Archive	Monthly	Dumps and purges old CDRs from the database.

CDR Ingestion

The ingestion job collects CDRs from the serving gateways, correlates them, and rates them. The job runs the Python tools in order.

1. Pull the 3GPP ASN.1 CDR files, and remove local files older than 60 days.
2. Convert the 3GPP ASN.1 files to XML, and import them.
3. Pull the OmniCore serving gateway CSV files.
4. Correlate the OmniCore records, and write them to Redis.
5. Drain Redis, rate each CDR, and store the result.

See [CDR Processing](#) for the pipeline detail and the age filters.

Scheduled TAP Export

The scheduler starts the daily export. The scheduler reads `config.export_schedule` on each tick, so a change applies without a restart. See the [Configuration Reference](#).

- The scheduler runs when `enabled` is `true`, the current UTC time reaches `at`, and the export has not run today.
- The scheduler enqueues one export job per target partner.
- The job queue runs one export at a time. This limit protects the shared database pool.
- When the queue drains, OmniRoam runs the command in `on_export_complete`.
- OmniRoam posts a Slack message to the configured webhook on completion or failure.

The scheduler does not run an extra export at boot. If the service starts after the daily time, the first export runs on the next day.

To run an export now, use the API. See the [HTTP API Reference](#).

File Transfer

OmniRoam moves files between the local store and the clearing house.

Upload

OmniRoam runs the upload command after each export, through `on_export_complete`. The upload command sends the new TAP files to the clearing house, and moves each sent file to the `OutputFiles_Sent` directory.

The upload script runs as the transfer user. The clearing house rejects the root user's key. Run the upload as the correct user, or the transfer will fail.

Inbound pull

The pull job retrieves inbound files from the clearing house into the inbound TAP directory. The pull job copies one file at a time, because the clearing house account does not allow a bulk transfer.

Prefix	File type
CD	An inbound live TAP file.
TD	An inbound test TAP file.
RC	A RAP file that rejects a live submission.
RT	A RAP file that rejects a test submission.

The pull job runs a daily incremental pull and a weekly full pull. The pull job runs as the transfer user.

Archiving

The archive job controls the growth of the CDR database. The archive job dumps the processed CDRs that are older than the retention window to a file, and then deletes those rows.

Schedule the archive job monthly. Confirm that the dump file is complete before you rely on the delete. Keep the retention window consistent with your database purge policy.

Sequence Counter Maintenance

The TAP file sequence number is a 5-digit field, with a range of 00001 to 99999. OmniRoam does not reset the counter automatically.

1. Monitor the CD and TD counters for each partner in counters.yaml.
2. Before a counter reaches 99999, agree a reset with the partner.

3. Set the counter back to `1` at the agreed time.
4. Back up `counters.yaml` with `config.yaml`.

See the [Configuration Reference](#).

Storage Maintenance

Monitor these stores for growth.

Store	Action
<code>OutputFiles</code>	Confirm the upload moves each sent file to <code>OutputFiles_Sent</code> . A growing pending count means the upload failed. Watch <code>tap3_files_pending</code> and <code>tap3_hours_since_last_file</code> .
InfluxDB	Set a retention policy. The <code>raw_cdr</code> measurement has high cardinality.
CDR database	Run the archive job on schedule.
Log files	The Python tools write a new timestamped log file per run in <code>/tmp</code> . Remove old log files.

See the [Metrics Reference](#) for the monitoring metrics.

Log Locations

The Python tools write one timestamped log file per run.

File pattern	Source
/tmp/import_CDR_Logger_Marben_*.log	The ASN.1 and Redis importers.
/tmp/import_CDR_Logger_OmniCore_*.log	The OmniCore importer.
/tmp/tap3_export_*.log	The legacy export tool.
/tmp/convert_cdr_logger_*.log	The 3GPP ASN.1 to XML converter.
/tmp/rate_creator_Logger_*.log	The rate provisioner.

The OmniRoam application logs through the system service manager. The upload script writes its status to a log file that the metrics collector reads.

-
- [Overview](#)

RAP Processing

This guide describes the Returned Account Procedure (RAP) files that partners return, and how OmniRoam alerts operators to them.

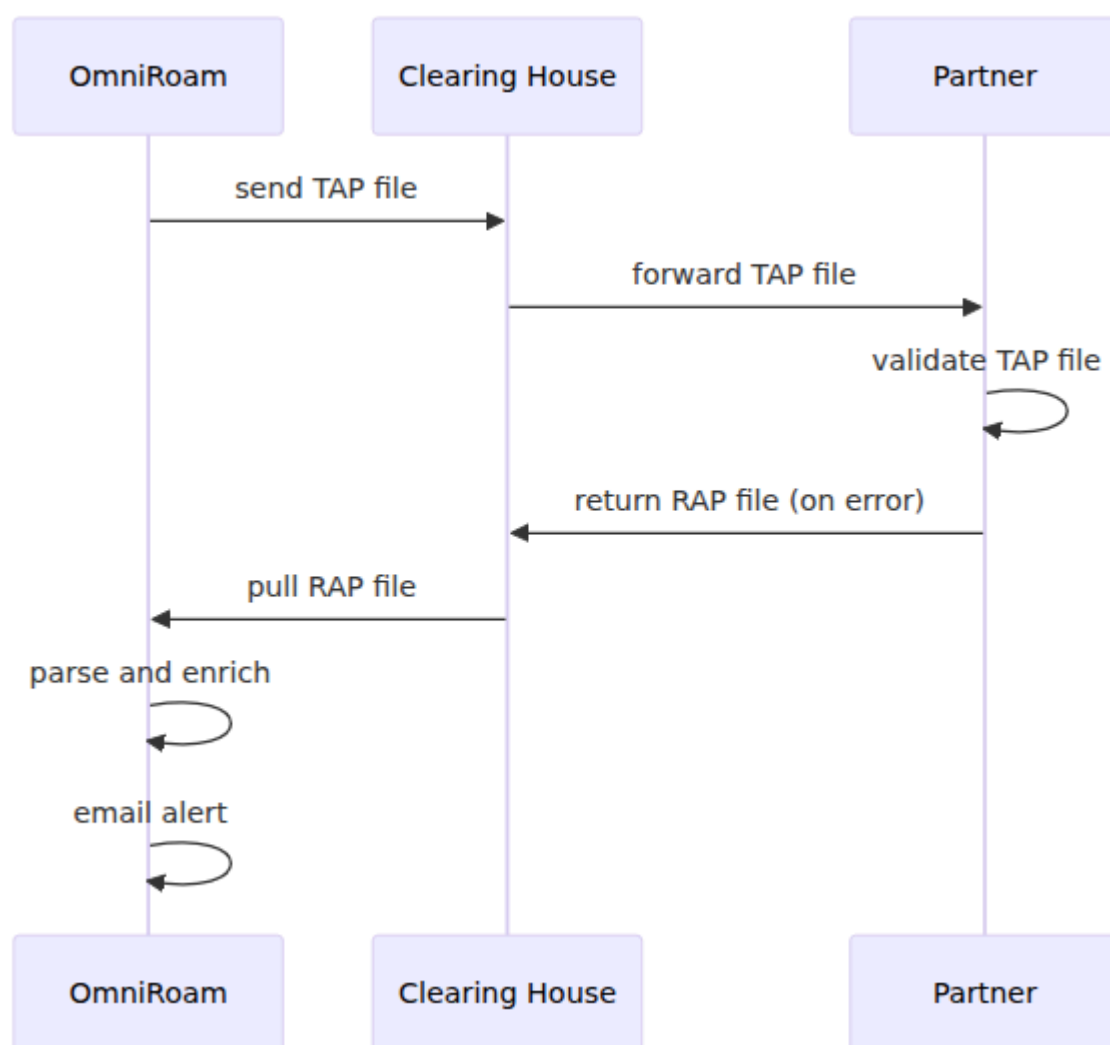
Table of Contents

- [What a RAP File Is](#)
- [RAP File Naming](#)
- [Return Types and Severity](#)
- [Error Codes](#)
- [RAP Monitoring and Alerts](#)
- [Reading a RAP File](#)

What a RAP File Is

A partner validates each TAP file that OmniRoam sends. If the partner finds errors, the partner returns a RAP file. The RAP file reports the errors in a specific TAP file. The RAP format follows [GSMA TD.32](#).

A RAP file is the messenger. The error codes describe faults in the original TAP file. OmniRoam reads the RAP file, adds a description to each error, and alerts operators.



RAP File Naming

A RAP file name has a fixed length of 17 characters. The name mirrors the TAP file name, but it uses a RAP prefix.

Prefix	Meaning
RC	A return against a live (Chargeable Data) TAP file.
RT	A return against a test (Test Data) TAP file.

The rest of the name holds the 5-character sender, the 5-character recipient, and the 5-digit sequence number. OmniRoam reads only files that start with RC or RT as RAP files.

Return Types and Severity

A RAP file holds one or more returns. Each return has a type and a severity.

Return type	Severity	Effect
Fatal return	Critical	The partner rejects the whole TAP file.
Severe return	Warning	The partner rejects the individual records, but accepts the rest of the file.
Missing return	Warning	The partner reports a gap in the file sequence.
Stop return	Information	The partner reports that it received no TAP file for 7 days or more.

Treat a fatal return first. A fatal return means the partner discarded the whole file, and you must correct the fault and resend.

Error Codes

OmniRoam adds a description to each error code in a RAP file. The codes come from [GSMA TD.57](#). The path identifiers come from TD.57 for the original TAP file, and from TD.32 for the RAP file.

The table below shows common error codes. See [TAP Validation](#) for the categories.

Code	Description
20	A required data item is missing.
50	A RAP encoding error in the ASN.1 structure.
107	An invalid structure.
200	The data is not in line with the roaming agreement.
250 to 257	A duplicate call event.
260	A call age error.
263	An invalid timestamp format.

RAP Monitoring and Alerts

The RAP monitor is a background process in the OmniRoam application. The monitor watches the inbound directories and alerts operators to a new RAP file.

Property	Value
Scan interval	Every 4 hours.
Directories	The inbound TAP directory and the RAP directory. A RAP file arrives with inbound TAP files.
Alert method	Email through the configured email provider.
Deduplication	OmniRoam records each alerted file. OmniRoam does not alert on the same file twice.

An email alert names the return type, the error count, and the file name. The alert also includes the batch information and the error codes.

Set the alert recipients and the email provider credentials in the application runtime configuration. Do not store the credentials in `config.yaml`.

To send a test alert for a known file, use `POST /api/rap_files/send_alert`. See the [HTTP API Reference](#).

Reading a RAP File

The OmniRoam API and the web interface read RAP files. A request to `GET /api/rap_files/:filename` returns the parsed and enriched RAP content as JSON.

OmniRoam extracts these fields from a RAP file:

Field	Description
<code>sender</code>	The TADIG code of the network that sent the RAP file.
<code>recipient</code>	The TADIG code of the network that receives the RAP file.
<code>roamingPartner</code>	The roaming partner TADIG code.
<code>rapFileSequenceNumber</code>	The RAP file sequence number.
<code>rapFileCreationTimeStamp</code>	The RAP file creation time.
<code>tapCurrency</code>	The settlement currency of the original TAP file.
<code>tapDecimalPlaces</code>	The decimal places of the original TAP file.
<code>returnDetailsCount</code>	The number of returns in the RAP file.
<code>totalSevereReturnValue</code>	The total value of the severe returns.

-
- Overview

TAP File Generation

This guide describes how OmniRoam builds TAP files, how it names them, and how it manages the sequence counters.

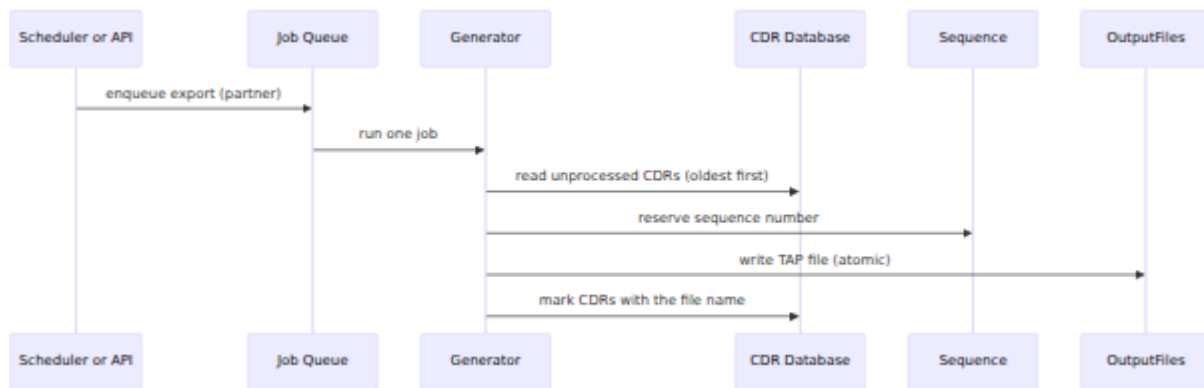
Table of Contents

- [Overview](#)
- [How a Generation Run Works](#)
- [File Naming](#)
- [Test Files and Live Files](#)
- [Sequence Counters](#)
- [TAP File Structure](#)
- [Charge Values](#)
- [NOTIFY Files](#)
- [Triggering an Export](#)

Overview

A TAP file is a binary file that reports rated roaming usage to one partner. OmniRoam encodes each TAP file as ASN.1 in the BER format, against the TAP 3.12 grammar. The format follows [GSMA TD.57](#).

How a Generation Run Works



1. The scheduler or an API request enqueues an export job for a partner.
2. The job queue runs one job at a time. This limit protects the shared database connection pool.
3. The generator reads the unprocessed CDRs for the partner, oldest first.
4. The generator splits the CDRs into files if the count is above the records-per-file limit.
5. The generator reserves a sequence number, encodes the file, and writes it atomically.
6. The generator marks each CDR in the database with the file name.

The generator writes each TAP file under a subdirectory named for the recipient TADIG, inside `tap_output_path`. The generator also writes a human-readable JSON copy to `tap_human_readable_output_path`.

File Naming

A TAP file name has a fixed length of 17 characters.

```
CD AAAAA BBBB 00011
```

Part	Length	Description
Prefix	2	CD for a live file, or TD for a test file.
Sender	5	Your TADIG code, from <code>batch_info.sender</code> .
Recipient	5	The partner's TADIG code, from <code>batch_info.recipient</code> .
Sequence	5	The zero-padded sequence number.

The name `CDAAAAABBBBB00011` means a live file, from `AAAAA`, to `BBBBB`, sequence `11`.

Test Files and Live Files

The `fileTypeIndicator` key in `batch_info` selects the file type.

- If `batch_info` does not contain `fileTypeIndicator`, OmniRoam produces a **live** file. The name uses the `CD` prefix and the `CD` sequence counter.
- If `batch_info` contains `fileTypeIndicator` with the value `T`, OmniRoam produces a **test** file. The name uses the `TD` prefix and the `TD` sequence counter.

Set `fileTypeIndicator` to `T` while you onboard a new partner. Remove the key to move the partner to live settlement. Use the value `T` only. A different value can produce a `TD` name with a `CD` sequence number, which breaks the counter alignment.

Sequence Counters

OmniRoam tracks a sequence number for each recipient TADIG and each file type. OmniRoam stores the counters in `counters.yaml`. See the [Configuration Reference](#).

- **CD** counts the live (Chargeable Data) files.

- **TD** counts the test (Test Data) files.

The sequence number increases by one for each file. The number does not increase for a retransmission. The TAP and RAP sequences are independent.

OmniRoam reserves a sequence number through a single process, and writes `counters.yaml` atomically. This method prevents two exports from using the same number. A failed write keeps the previous counter file intact.

The sequence number is a 5-digit field with a range of `00001` to `99999`.

OmniRoam does not reset the counter automatically. See the [Operations Guide](#) for the reset procedure.

TAP File Structure

A TAP file is a `transferBatch` structure. The batch holds control information and a list of call event records. OmniRoam produces GPRS (data) records only.

Section	Contents
<code>batchControlInfo</code>	The sender, the recipient, the file sequence number, the file timestamps, and the version numbers.
<code>accountingInfo</code>	The local currency, the settlement currency, the decimal places, and the exchange rate information.
<code>networkInfo</code>	The UTC offset table and the recording entity list (the packet gateway and the serving gateway).
<code>callEventDetails</code>	The list of GPRS call records.
<code>auditControlInfo</code>	The earliest and latest call timestamps, the total charge, and the record count.

Each GPRS call record holds the fields below.

Field	Description
Chargeable subscriber	The IMSI, and the MSISDN when present.
PDP address	The subscriber IP address. OmniRoam prefers IPv6.
Access point name	The network identifier and the operator identifier of the APN.
Call event start timestamp	The charging start time, with the UTC offset for the serving location.
Total call event duration	The session duration in seconds.
Charging ID	The session charging identifier.
Location information	The recording entities, the location area, the cell identifier, and the optional serving billing identifier.
Data volumes	The incoming and outgoing byte counts.
Charge information	The charge, the chargeable units, the charged units, and the call type levels.

Charge Values

TAP files store money as an integer. The reader divides the integer by `10tapDecimalPlaces` to get the decimal value.

OmniRoam scales each charge by 1000 to produce the integer. This scale matches `tapDecimalPlaces` of 3. Set `tapDecimalPlaces` to 3 for each partner. A different value produces a charge integer that does not match the declared decimal places.

The `roundingAction` setting controls how OmniRoam rounds the scaled charge:

- `Up` rounds the value up.
- `Down` rounds the value down.
- `Simple` rounds the value to the nearest integer.

NOTIFY Files

A partner expects a file even when there is no usage. When a partner has no unprocessed CDRs, OmniRoam writes a NOTIFY file. A NOTIFY file uses the same name format as a TAP file, but it holds a `notification` record instead of a batch of call events.

Triggering an Export

You can start an export in three ways.

Method	Action
Scheduler	OmniRoam runs the export automatically at the configured time. See the Operations Guide .
API	Send <code>POST /api/tap_export</code> for one partner, or <code>POST /api/export_schedule</code> for every partner. See the HTTP API Reference .
Command line	Run the legacy export tool with the partner name. Use this method as a fallback only.

-
- [Overview](#)

Troubleshooting

This guide lists common problems and their resolutions. Each entry gives the symptoms, the possible causes, and the steps to resolve the problem.

Table of Contents

- [CDRs Do Not Reach a Partner](#)
- [TAP Files Are Not Uploaded](#)
- [Wrong Sequence Numbers](#)
- [A Partner Returns a RAP File](#)
- [Missing Location in TAP Files](#)
- [Wrong Charge Values](#)
- [No Metrics or Analytics](#)

CDRs Do Not Reach a Partner

Symptoms: A partner reports missing usage. The CDRs are in the database but not in a TAP file.

Possible causes:

- The CDR IMSI matches no partner prefix.
- The CDR is more than 30 days old, so ingestion removed it.
- The export did not run for the partner.

Resolution:

1. Confirm the CDR IMSI matches a prefix in `imsi_prefixes` for the partner.
2. Confirm the CDR age is within 30 days. Ingest and rate CDRs promptly. See [CDR Processing](#).
3. Confirm the export ran. Check the export jobs through the [HTTP API Reference](#).

TAP Files Are Not Uploaded

Symptoms: The pending file count grows. The metric `tap3_files_pending` is high.

Possible causes:

- The upload command in `on_export_complete` is blank or wrong.
- The upload runs as the wrong user, so the clearing house rejects the key.
- The clearing house is not reachable.

Resolution:

1. Confirm the `on_export_complete` command is set. See the [Configuration Reference](#).
2. Confirm the upload runs as the transfer user, not root.
3. Check the upload log and the metric `tap3_upload_status`.

Wrong Sequence Numbers

Symptoms: A partner reports a sequence gap or a duplicate sequence number.

Possible causes:

- The `counters.yaml` file was restored from an old backup.
- A counter reached `99999`.
- The `fileTypeIndicator` value is set to a value other than `T`.

Resolution:

1. Check the `CD` and `TD` counters for the recipient TADIG in `counters.yaml`.
2. If a counter is near `99999`, follow the reset procedure in the [Operations Guide](#).
3. Set `fileTypeIndicator` to `T` for a test file, or remove the key for a live file. See [TAP File Generation](#).

A Partner Returns a RAP File

Symptoms: The RAP monitor sends an email alert. A RAP file is in the inbound directory.

Possible causes:

- The TAP file failed the partner's validation.

Resolution:

1. Read the return type. Treat a fatal return first, because the partner rejected the whole file.
2. Read the error codes in the alert or through `GET /api/rap_files/:filename`.
3. Correct the fault, and resend the file. See [RAP Processing](#).

Missing Location in TAP Files

Symptoms: A TAP file has no serving billing identifier or location description.

Possible causes:

- The CDR TAC matches no `tac_config` template.
- The template has no `servingBid`.

Resolution:

1. Confirm the CDR TAC is in a `tac_list`. See the [Configuration Reference](#).
2. Confirm the template has a `servingBid` and a `servingLocationDescription`.

Wrong Charge Values

Symptoms: The charge values do not match the expected settlement amount.

Possible causes:

- The `unit_price` or `unit_bytes` value is wrong.
- The `tapDecimalPlaces` value is not `3`.
- The `roundingAction` value is wrong.

Resolution:

1. Confirm the `unit_price` and `unit_bytes` values. See the [Configuration Reference](#).
2. Set `tapDecimalPlaces` to `3`. See [TAP File Generation](#).
3. Confirm the `roundingAction` value.
4. Reload the rates into the Charging Engine after any rate change.

No Metrics or Analytics

Symptoms: The InfluxDB analytics are empty. A Prometheus metric is absent.

Possible causes:

- The `influx_db` section is incomplete.
- A dependency service is down.

Resolution:

1. Confirm all four `influx_db` values are set. OmniRoam skips the write if any value is missing. See the [Configuration Reference](#).
2. Check the metric `tap3_service_up` for a service that is down. See the [Metrics Reference](#).

-
- [Overview](#)

TAP Validation

OmniRoam validates TAP files against the checkable rules in [GSMA TD.57](#) and [GSMA TD.32](#). You can validate an inbound TAP file or an outbound TAP file through the API.

Table of Contents

- [How to Validate a File](#)
- [Validation Checks](#)
- [Error Severity](#)
- [Error Code Categories](#)
- [Limits of Validation](#)

How to Validate a File

Use the API to validate a TAP file.

Endpoint	Purpose
<code>GET /api/tap_validate_in/:filename</code>	Validate an inbound TAP file.
<code>GET /api/tap_validate_out/:filename</code>	Validate an outbound TAP file.

The response reports whether the file is valid, and it counts the findings by severity. Each finding holds the error code, the severity, the location, and a message. See the [HTTP API Reference](#).

Validation Checks

OmniRoam runs four groups of checks over a decoded TAP file.

Group	Checks
Structure	The five mandatory sections are present and not blank: <code>batchControlInfo</code> , <code>accountingInfo</code> , <code>networkInfo</code> , <code>auditControlInfo</code> , and <code>callEventDetails</code> .
Batch control	The <code>sender</code> and <code>recipient</code> are 5 characters. The <code>fileSequenceNumber</code> is 5 characters. The <code>specificationVersionNumber</code> is 3.
Accounting	The <code>localCurrency</code> and <code>tapCurrency</code> are 3-letter ISO 4217 codes. The <code>tapDecimalPlaces</code> is a non-negative integer.
Audit arithmetic	The record count equals the number of call events. The total charge, total tax, and total discount equal the sums from the records.

Error Severity

OmniRoam reports each finding with a severity. The file is valid only when there are no fatal findings.

Severity	Effect
Fatal	The whole file is rejected.
Severe	The individual record is rejected. The rest of the file is accepted.
Warning	The record is billable. The finding is informational.

Error Code Categories

TD.57 groups the error codes into numeric ranges.

Range	Category
10 to 19	Format or syntax error.
20 to 29	Value out of range.
30 to 49	Invalid group structure.
50 to 59	General ASN.1 or encoding error.
100 to 199	Cross-field inconsistency.
200 to 249	Specific error, such as a duplicate, an IOT error, or old data.
9000 to 9999	A bilateral, proprietary error agreed between two partners.

Limits of Validation

OmniRoam validates the rules that it can check from the file content alone. Some TD.57 rules need external reference data, and OmniRoam does not check them.

- TADIG code lists.
- Inter-operator tariff and roaming agreement rates.
- Duplicate history across earlier files.
- Full ISO 4217 currency and E.164 number lookups.

Use a partner acknowledgement or a RAP file to confirm the settlement result. See [RAP Processing](#).

-
- [Overview](#)

