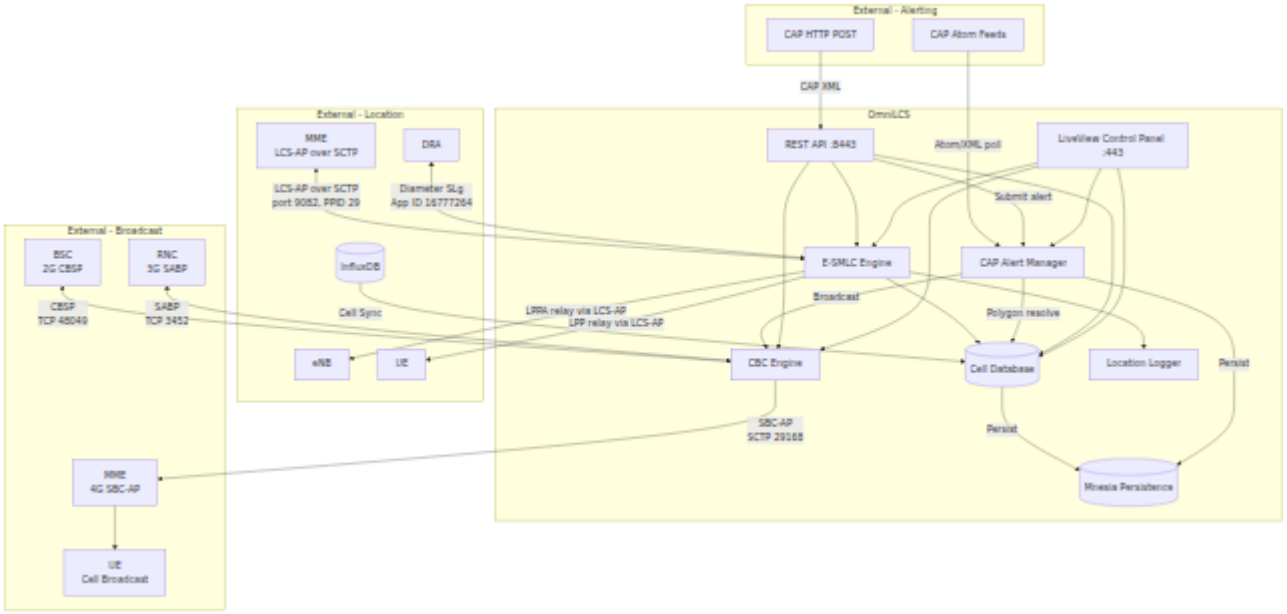
OmniLCS Operations Guide

OmniLCS is an integrated LTE/GSM Location and Cell Broadcast platform. It combines an **E-SMLC** (Evolved Serving Mobile Location Centre) for UE positioning with a **CBC** (Cell Broadcast Centre) for public warning and commercial broadcast services, all within a single Elixir/OTP application.

Architecture Overview



Feature Overview

E-SMLC -- Location Services

- **Positioning Methods**: Cell ID, Enhanced Cell ID (E-CID), GNSS/A-GPS, OTDOA
- **LCS-AP over SCTP (SLs Interface)**: Communicates with the MME using native LCS-AP per 3GPP TS 29.171, port 9082, PPID 29

- **LPPA/LPP Relay**: Sends positioning protocol messages to eNBs and UEs through the MME via LCS-AP Connection Oriented Information
- **Cell Database**: Stores cell positions for Cell ID and OTDOA positioning. Supports import from Huawei U2020 XLSX (GSM/UMTS/LTE/NR), vendor-specific CSV, JSON, and InfluxDB sync. Persisted to Mnesia for survival across restarts
- **OTDOA Multilateration**: Computes UE position from RSTD measurements using iterative least-squares
- **Location Logging**: Persists every location fix to CSV and in-memory history. Optional InfluxDB logging when configured
- **Subscriber Tracking**: Periodic location polling per IMSI with configurable interval and method. Results stored in Mnesia with KML/CSV export
- **Virtual Drive Test**: Multi-IMSI campaigns combining GNSS positioning with E-CID signal measurements (RSRP/RSRQ). Campaign-level management with real-time monitoring, InfluxDB logging, and KML/CSV export with signal quality data

GMLC -- Gateway Mobile Location Centre

- **Le Interface**: Diameter interface for external LCS clients (PSAPs, fleet management, lawful intercept) per 3GPP TS 29.172
- **Client Authorization**: Per-client access control with type matching and configurable rate limits
- **Periodic Location**: Deferred sessions performing position fixes at configurable intervals with InfluxDB logging and Diameter LRR delivery
- **Triggered Location (Geo-fence)**: Area event subscriptions with entering/leaving/being-inside triggers against circular or polygon areas
- **Location Report Delivery**: Sends Diameter Location-Report-Request (LRR) messages to originating clients for each deferred fix

CBC -- Cell Broadcast

- **2G CBSP**: TCP server on port 48049 accepting connections from BSCs
- **3G SABP**: TCP on port 3452 (Iu-BC interface, 3GPP TS 25.419, transport per TS 25.414 §7.1.3.3). CBC connects outbound to configured RNC peers; also listens for inbound RNC connections (Failure/Restart Indications)

- **4G SBC-AP**: SCTP client connecting to MME peers on port 29168
- **Multi-Language Broadcasts**: Send the same alert in multiple languages simultaneously, each with its own CBS Data Coding Scheme and Message Code per 3GPP TS 23.038
- **Broadcast Updates**: Update an active broadcast by incrementing the Update Number. Automatically sends Stop-Warning for the old serial before the new Write-Replace
- **Message Formatting**: GSM 7-bit and UCS-2 encoding, CBS page assembly, serial number construction
- **Warning Types**: ETWS support (earthquake, tsunami, test) with emergency alert and popup activation
- **Broadcast State Management**: Active broadcast tracking with persistence across restarts, response timeout monitoring
- **Mnesia Persistence**: Cell database and CAP alert state persisted to Mnesia disc_copies tables, surviving application restarts without re-import
- **PWS Procedures**: Handles PWS-Restart-Indication and PWS-Failure-Indication from MMEs
- **CAP Alert Ingestion**: Accepts Common Alerting Protocol (CAP) v1.2 XML from external alerting authorities via HTTP POST or Atom feed polling, resolves polygon warning areas to TACs/LACs, and triggers cell broadcasts with optional operator approval

Control Interfaces

- **REST API**: HTTPS on port 8443 with endpoints for location requests, cell management, and system status
- **LiveView Control Panel**: HTTPS on port 443 with real-time dashboard, location testing, cell database management, Diameter peer monitoring, and CBC broadcast composition

Documentation Structure

Document	Description
Configuration Reference	All configuration parameters with tables and examples
Cell Broadcast Operations	CBSP (2G), SABP (3G), and SBC-AP (4G) operations, message formatting, troubleshooting
CAP Alert Operations	CAP v1.2 ingestion, polygon resolution, approval workflow, feed polling
Location Services	E-SMLC positioning methods, LCS-AP interface, cell database, OTDOA
Subscriber Tracking & Drive Test	Periodic tracking, virtual drive test campaigns, RSRP/RSRQ measurements, KML/CSV export
GMLC & Le Interface	External LCS client access, periodic and geo-fence location, InfluxDB logging
REST API Reference	All API endpoints with request/response examples
Web Interface Guide	LiveView control panel pages and features

Interfaces Summary

Interface	Protocol	Transport	Port	Direction	3GPP Reference
SLs (E-SMLC ↔ MME)	LCS-AP	SCTP	9082	OmniLCS connects to MMEs	TS 29.171
SLg/Le (GMLC ↔ LCS Clients/DRA)	Diameter	SCTP	3868	Inbound from LCS clients, outbound to DRA	TS 29.172
CBSP	CBSP	TCP	48049	BSCs connect to OmniLCS	TS 48.049
SABP (Iu-BC)	SABP	TCP	3452	Outbound to RNCs / Inbound from RNCs	TS 25.414 §7.1.3.3
SBC-AP	SBC-AP	SCTP	29168	OmniLCS connects to MMEs	TS 29.168
REST API	HTTPS	TCP	8443	Clients connect to OmniLCS	--
Control Panel	HTTPS	TCP	443	Browsers connect to OmniLCS	--

Interface	Protocol	Transport	Port	Direction	3GPP Reference
Cell Sync	HTTP	TCP	8086	OmniLCS queries InfluxDB	--

Supervision Tree

OmniLCS starts the following processes under a one-for-one supervisor:

1. **OmniLcs.Persistence** -- Mnesia initialization and disc-backed persistence for cell database and CAP alerts
2. **OmniLcs.Context** -- ETS table initialization (cell_database, location_session, pending_transactions); loads persisted cells from Mnesia on startup
3. **OmniLcs.InfluxDb** -- InfluxDB connection pool
4. **OmniLcs.Esmlc.LocationLogger** -- Location fix logging to CSV and ETS
5. **Task.Supervisor** -- Async task execution
6. **OmniLcs.Sls.SctpTransport** -- SLs LCS-AP SCTP client connections to MMEs
7. **DiameterEx.Supervisor** -- Diameter service and peer management (SLg and other Diameter interfaces)
8. **OmniLcs.Esmlc.CellSync** -- Periodic InfluxDB cell synchronization
9. **OmniLcs.Cbc.CbspConnectionSupervisor** -- DynamicSupervisor for 2G CBSP TCP connections
10. **OmniLcs.Cbc.CbspTransport** -- CBSP TCP listener (port 48049)
11. **OmniLcs.Cbc.SabpConnectionSupervisor** -- DynamicSupervisor for 3G SABP TCP connections
12. **OmniLcs.Cbc.SabpTransport** -- SABP TCP listener (port 3452)
13. **OmniLcs.Cbc.Engine** -- CBC broadcast state management (2G, 3G, and 4G)
14. **OmniLcs.Cbc.SctpTransport** -- SBC-AP SCTP client connections to MMEs

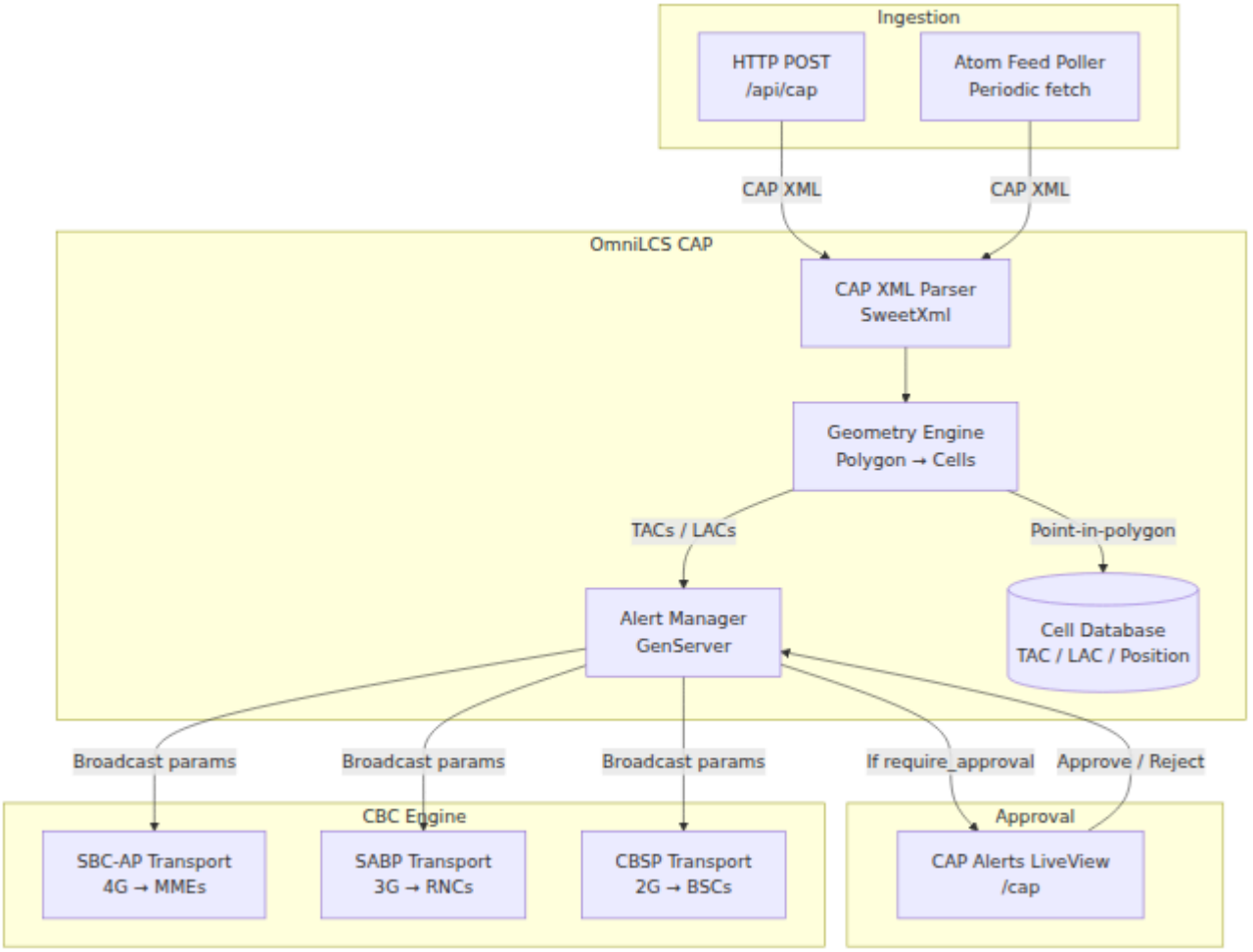
15. **OmniLcs.Cap.AlertManager** -- CAP alert lifecycle management (parse, resolve, approve, broadcast); persisted to Mnesia
16. **OmniLcs.Cap.FeedPoller** -- Periodic CAP Atom feed polling
17. **OmniLcs.Gmlc.ClientRegistry** -- Authorized external LCS client management
18. **OmniLcs.Gmlc.SessionSupervisor** -- DynamicSupervisor for periodic and triggered location sessions
19. **OmniLcs.Tracking.SessionSupervisor** -- DynamicSupervisor for subscriber tracking watch sessions
20. **OmniLcs.DriveTest.CampaignSupervisor** -- DynamicSupervisor for virtual drive test campaigns

CAP Alert Operations Guide

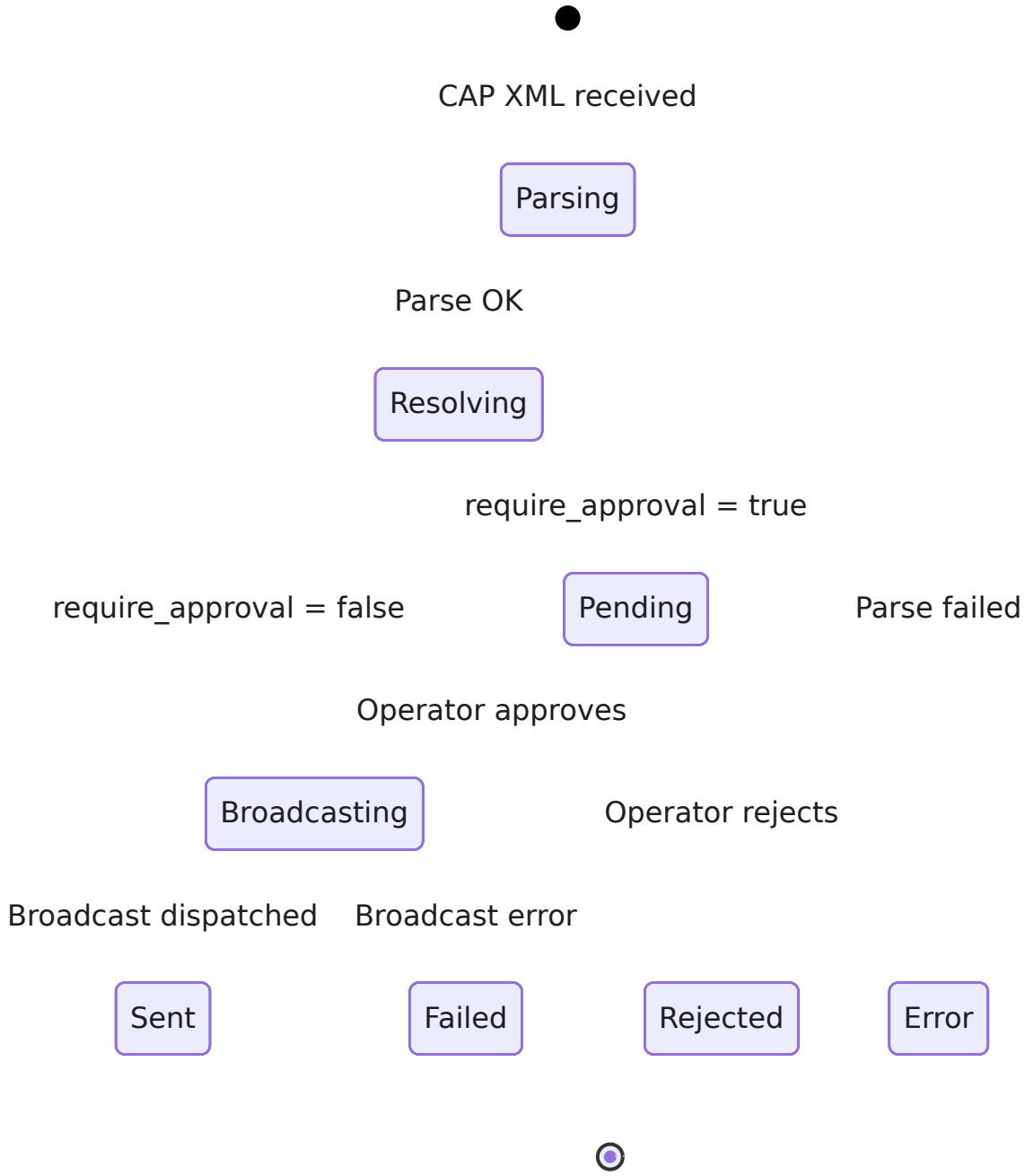
OmniLCS accepts **Common Alerting Protocol (CAP) v1.2** XML messages from external alerting authorities (weather services, emergency management agencies, etc.), resolves geographic warning areas to the network's cell infrastructure, and triggers cell broadcasts via the existing CBC engine across 4G (SBC-AP), 3G (SABP), and 2G (CBSP) interfaces.

Two ingestion methods are supported: **HTTP POST** (push) for direct integration and **Atom feed polling** (pull) for monitoring standard CAP feed sources. A configurable **manual approval step** allows operators to preview alerts before broadcast.

Architecture



Alert Lifecycle



When a CAP alert is received (via HTTP POST or feed poll):

1. The **CAP XML parser** extracts the alert envelope, info blocks, warning areas (polygons), and CB parameters
2. The **geometry engine** matches each polygon against the cell database using ray-casting (point-in-polygon), resolving warning areas to lists of TACs (4G), SAIs (3G), and LACs (2G)
3. If `require_approval` is `true`, the alert is queued as **pending** and appears in the operator LiveView UI for preview and approval

4. If `require_approval` is `false`, the alert is auto-approved and immediately dispatched to the CBC engine
5. On broadcast, the CBC engine sends Write-Replace-Warning-Request (4G SBC-AP), Write-Replace (3G SABP), and/or WRITE-REPLACE (2G CBSP) to all connected MMEs, RNCs, and BSCs

Configuration

CAP Settings

```
config :omnilcs, :cap,
  # Require operator approval before broadcasting
  require_approval: true,

  # PLMN identity for broadcast messages
  plmn: %{mcc: "001", mnc: "01"},

  # Use cell coverage radius for polygon matching (vs center-point
only)
  coverage_aware: false,

  # Atom feed sources to poll (empty = no polling)
  feeds: []
```

CAP Parameters

Parameter	Type	Required	Default	Description
`require_approval`	boolean	No	`true`	When `true`, alerts are queued for operator approval before broadcast. When `false`, alerts are auto-approved and broadcast immediately.
`plmn`	map	No	`%{mcc: "001", mnc: "01"}`	PLMN identity (MCC/MNC) used in broadcast messages. Must match the serving network.
`plmn.mcc`	string	Yes	`"001"`	Mobile Country Code (3 digits).
`plmn.mnc`	string	Yes	`"01"`	Mobile Network Code (2-3 digits).
`coverage_aware`	boolean	No	`false`	When `true`, polygon matching uses each cell's coverage radius to determine intersection rather than just the center point. Useful for large

Parameter	Type	Required	Default	Description
				cells at polygon edges.
`feeds`	list	No	`[]`	List of CAP Atom feed configurations to poll. See Feed Configuration.

Feed Configuration

Each entry in the `feeds` list defines a CAP Atom feed source to monitor.

```
config :omnilcs, :cap,
  feeds: [
    %{url: "https://alerts.weather.gov/cap/us.php?x=1",
poll_interval_seconds: 60},
    %{url: "https://feeds.meteoalarm.org/api/v1/warnings/atom",
poll_interval_seconds: 120}
  ]
```

Parameter	Type	Required	Default	Description
`url`	string	Yes	--	URL of the CAP Atom feed. Must return Atom XML (RFC 4287) with embedded CAP alert entries.
`poll_interval_seconds`	integer	No	`60`	Seconds between feed polls. Lower values provide faster alert ingestion but increase network traffic.

The feed poller tracks seen alert identifiers to avoid reprocessing. Each new `<entry>` in the feed has its embedded `<alert>` XML extracted and submitted to the Alert Manager.

Configuration Example: Full CAP Setup

```
config :omnilcs, :cap,
  require_approval: true,
  plmn: %{mcc: "001", mnc: "01"},
  coverage_aware: true,
  feeds: [
    %{url: "https://alerts.weather.gov/cap/us.php?x=1",
poll_interval_seconds: 60}
  ]
```

How it works: CAP alerts from the NWS weather feed are checked every 60 seconds. New alerts are parsed and their polygon warning areas are matched against the cell database using coverage-aware intersection. Matched alerts appear in the CAP Alerts LiveView page for operator review. The operator can preview the alert details (message text, severity, matched cells, TACs/LACs), then approve to broadcast or reject to discard.

Cell Database: TAC, LAC, and RAT Fields

CAP polygon resolution depends on cells in the database having **TAC** (4G), **LAC** (2G and 3G), **SAC** (3G), and **RAT** fields populated. These fields determine which broadcast interface is used for each cell.

Cell Fields for CAP

Field	Type	Description
`tac`	integer	Tracking Area Code. Used for 4G SBC-AP broadcasts. Cells with a TAC are included in the TAI list sent to MMEs.
`lac`	integer	Location Area Code. Used for 2G CBSP and 3G SABP broadcasts. Cells with a LAC are included in the cell list sent to BSCs and the SAI list sent to RNCs.
`rat`	string	Radio Access Technology: `"4g"`, `"3g"`, or `"2g"`. Informational; the presence of `tac` or `lac` determines broadcast targeting.

These fields can be set via:

- **REST API** when creating or updating cells (`POST /api/cells`, `PUT /api/cells/:id`)
- **JSON import** (`priv/cells.json`) with `"tac"`, `"lac"`, and `"rat"` fields per cell

- **Manual entry** in the Cell Database LiveView page

When syncing cells from InfluxDB, existing TAC/LAC/RAT values are preserved (InfluxDB does not provide these fields, so they are not overwritten with `nil`).

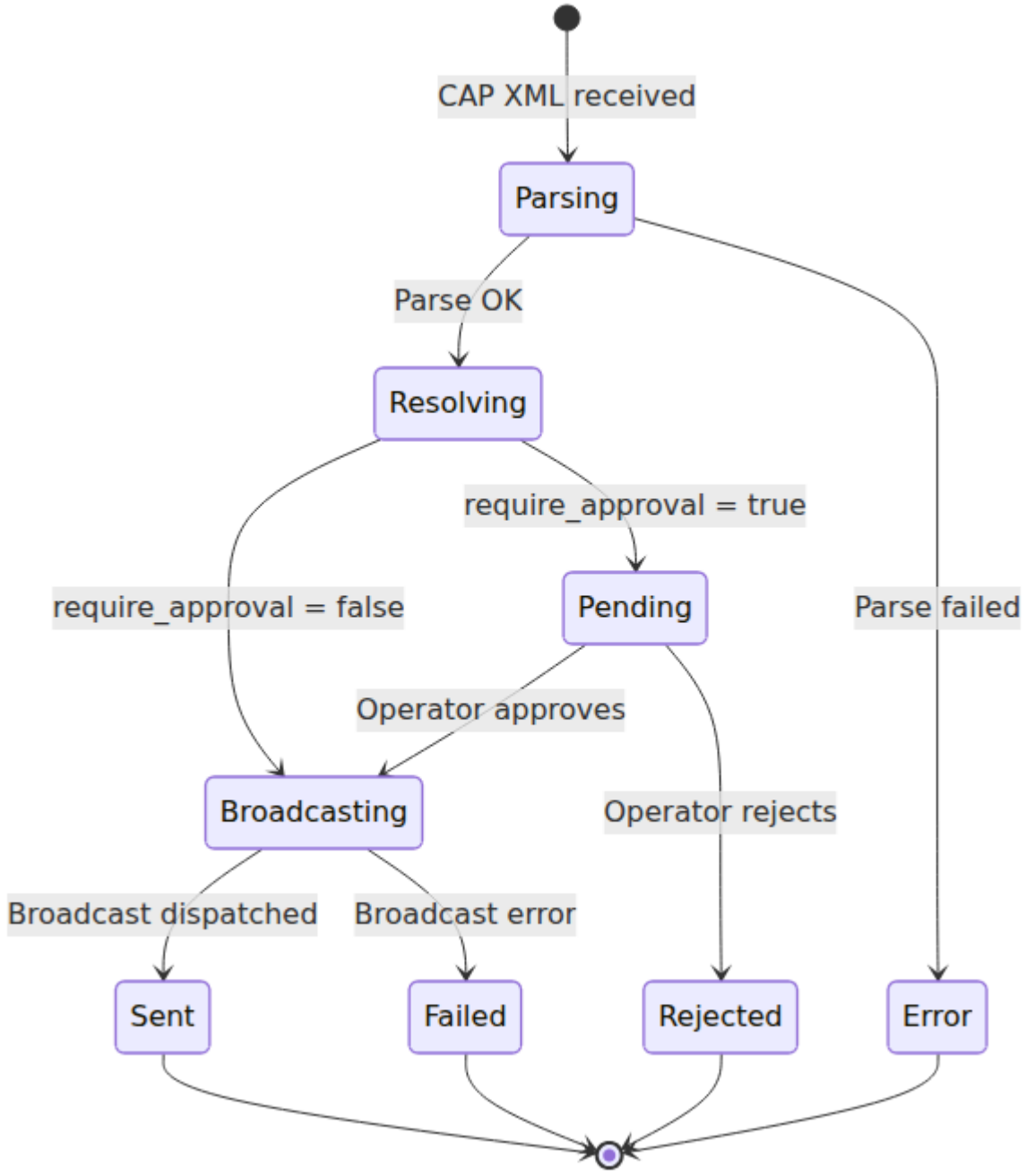
Cell JSON Example

```
[
  {
    "cell_id": "eNB-001-cell-01",
    "latitude": 40.7128,
    "longitude": -74.0060,
    "pci": 100,
    "earfcn": 1300,
    "radius": 500,
    "tac": 100,
    "lac": null,
    "rat": "4g"
  },
  {
    "cell_id": "BTS-001-cell-01",
    "latitude": 40.7130,
    "longitude": -74.0065,
    "pci": null,
    "earfcn": null,
    "radius": 2000,
    "tac": null,
    "lac": 5001,
    "rat": "2g"
  }
]
```

Polygon Resolution

When a CAP alert contains polygon warning areas, the geometry engine determines which cells fall inside each polygon.

How Polygon Matching Works



Standard mode (`coverage_aware: false`): A cell matches if its center point (latitude/longitude) falls inside the polygon, using the ray-casting algorithm.

Coverage-aware mode (`coverage_aware: true`): A cell matches if its coverage area (modeled as a circle with the cell's `radius` field) intersects the polygon. This catches cells whose center is just outside the polygon but whose coverage extends into it.

CAP Polygon Format

CAP v1.2 defines polygons as space-separated `lat,lon` pairs. The first and last points should be identical (closed polygon):

```
38.47,-120.14 38.34,-119.95 38.52,-119.74 38.62,-119.89
38.47,-120.14
```

CAP XML Format

The parser handles CAP v1.2 per the OASIS standard. Key elements extracted:

Alert Envelope

Element	Description
`<identifier>`	Unique alert identifier
`<sender>`	Alert originator
`<sent>`	Timestamp of alert issuance
`<status>`	`Actual`, `Exercise`, `System`, `Test`, `Draft`
`<msgType>`	`Alert`, `Update`, `Cancel`, `Ack`, `Error`
`<scope>`	`Public`, `Restricted`, `Private`

Info Block

Each `<info>` block contains the alert content for a specific language/audience:

Element	Description
`<category>`	Alert category (Geo, Met, Safety, Security, etc.)
`<event>`	Event type description (e.g., "Tornado Warning")
`<urgency>`	Immediate, Expected, Future, Past, Unknown
`<severity>`	Extreme, Severe, Moderate, Minor, Unknown
`<certainty>`	Observed, Likely, Possible, Unlikely, Unknown
`<headline>`	Short alert headline
`<description>`	Full alert description text (used as broadcast message)
`<instruction>`	Recommended action

CB Parameters

Cell broadcast parameters are extracted from `<parameter>` elements within each `<info>` block:

Parameter Name	Description
`CBMessageIdentifier`	16-bit CB message ID for the broadcast
`CBRepetitionInterval`	Seconds between broadcast repetitions
`CBNumberOfBroadcasts`	Total number of times to broadcast

If these parameters are not present in the CAP XML, defaults are used (message ID `0x1112`, repetition period 30 seconds, 10 broadcasts).

Area Block

Each `<area>` within an `<info>` can contain:

Element	Description
`<areaDesc>`	Human-readable area description
`<polygon>`	Space-separated `lat,lon` coordinate pairs defining the warning area
`<circle>`	Center point and radius (for future use)

REST API

POST /api/cap

Submit a CAP XML alert for processing.

Request Body

```
{
  "xml": "<alert
xmlns=\"urn:oasis:names:tc:emergency:cap:1.2\">...</alert>"
}
```

Parameter	Type	Required	Description
`xml`	string	Yes	Complete CAP v1.2 XML alert document

Response (201) -- Alert submitted successfully

```
{
  "status": "ok",
  "data": {
    "id": "a1b2c3d4-e5f6-...",
    "status": "pending",
    "source": "http_post",
    "received_at": "2025-01-15T10:30:00Z",
    "matched_cells": 42,
    "tacs": [100, 101, 102],
    "lacs": [5001, 5002],
    "mcc": "001",
    "mnc": "01",
    "broadcast_params": {
      "message_id": 4370,
      "repetition_period": 30,
      "num_broadcasts": 10,
      "message_text": "Tornado Warning for Springfield County...",
      "event": "Tornado Warning",
      "severity": "Extreme",
      "urgency": "Immediate"
    }
  }
}
```


The `status` field will be `"pending"` if `require_approval` is `true`, or `"sent"` if auto-approved.

Error Responses

Status	Reason
400	`"xml field is required"`
422	Parse error details

GET /api/cap

List all alerts across all states.

Response (200)

```
{
  "status": "ok",
  "data": {
    "pending": [...],
    "active": [...],
    "history": [...]
  }
}
```

Each array contains alert objects in the same format as the POST response.

GET /api/cap/:id

Get a single alert by ID.

Path Parameters

Parameter	Type	Description
`id`	string	Alert UUID

Response (200)

Returns the alert object.

Error Response

Status	Reason
404	`"Alert not found: <id>"`

PUT /api/cap/:id

Approve or reject a pending alert.

Request Body

```
{
  "action": "approve",
  "operator": "operator1"
}
```

Parameter	Type	Required	Description
`action`	string	Yes	`"approve"` or `"reject"`
`operator`	string	No	Operator name for audit trail. Defaults to `"unknown"`.

Response (200)

Returns the updated alert object with new status (`"sent"`, `"broadcasting"`, or `"rejected"`).

Error Responses

Status	Reason
400	`"action must be 'approve' or 'reject'"`
404	`"Alert not found: <id>"`

Web Interface: CAP Alerts Page

Path: `/cap` **Refresh**: Every 3 seconds + real-time PubSub updates

The CAP Alerts page provides the operator approval workflow and broadcast monitoring interface.

Statistics Cards

Four summary cards across the top:

Card	Description
Pending Approval	Number of alerts awaiting operator action (amber when > 0)
Active Broadcasts	Number of currently broadcasting alerts
Total Processed	Total alerts in history (sent + rejected + failed)
Approval Mode	Current mode: "Manual" or "Auto"

Pending Alerts Panel

Only visible when `require_approval` is `true`. Shows alerts awaiting operator approval.

Column	Description
TIME	When the alert was received (HH:MM:SS)
EVENT	Alert event type (e.g., "Tornado Warning")
SEVERITY	Alert severity level
CELLS	Number of cells matched by polygon resolution
STATUS	Pending badge (amber)
ACTIONS	Preview, Approve, and Reject buttons

Preview expands the row to show:

Detail	Description
Description	Full alert description text (the broadcast message content)
Source	How the alert was received (`http_post` or `feed_poll`)
TACs	Matched Tracking Area Codes for 4G broadcast
LACs	Matched Location Area Codes for 2G and 3G broadcast
Message ID	CB message identifier
PLMN	MCC/MNC used for broadcast

Approve triggers immediate broadcast via the CBC engine to all connected MMEs (4G), RNCs (3G), and BSCs (2G).

Reject moves the alert to history with `:rejected` status.

Active Broadcasts Panel

Shows alerts currently being broadcast:

Column	Description
EVENT	Alert event type
MSG ID	CB message identifier
TACs	Targeted Tracking Area Codes
STARTED	Broadcast start time
STATUS	Broadcasting (blue) or Sent (green) badge

Alert History Panel

Shows sent, rejected, and failed alerts (last 200):

Column	Description
TIME	Sent or received timestamp
EVENT	Alert event type
SEVERITY	Alert severity level
CELLS	Matched cell count
TACs/LACs	Targeted TACs and LACs
STATUS	Sent (green), Rejected (red), or Failed (red) badge

Real-Time Updates

The page subscribes to the `cap:alerts` PubSub topic. When a new alert arrives or an alert status changes, the page auto-refreshes. A toast notification appears when a new pending alert is received ("New CAP alert received — awaiting approval").

Alert State Persistence

Alert state is persisted to `priv/cap_alerts.json` in JSON format, following the same pattern as the CBC Engine's `priv/active_broadcasts.json`. On startup, the Alert Manager reloads this file to restore pending and active alerts.

The history is capped at **200 entries** to prevent unbounded growth.

Atom Feed Polling

The feed poller supports standard CAP Atom feeds as used by national weather services and alerting authorities. Feed format follows RFC 4287 (Atom Syndication) with CAP alert entries.

Expected Feed Format

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <entry>
    <id>urn:oid:2.49.0.1.840.0.abc123</id>
    <title>Tornado Warning</title>
    <content type="text/xml">
      <alert xmlns="urn:oasis:names:tc:emergency:cap:1.2">
        <!-- Full CAP alert XML -->
      </alert>
    </content>
  </entry>
</feed>
```

Deduplication

The poller tracks the `<id>` of every entry it has seen. Entries with previously-seen IDs are skipped. The dedup set is maintained in memory and resets on application restart.

Error Handling

Feed fetch failures (network errors, HTTP errors, XML parse errors) are logged as warnings and do not affect subsequent poll cycles. The poller continues on its configured interval regardless of individual fetch outcomes.

Supervision

The CAP subsystem adds two processes to the OmniLCS supervisor:

Process	Description
`OmniLcs.Cap.AlertManager`	Alert lifecycle GenServer. Manages pending/active/history state, polygon resolution, and broadcast dispatch.
`OmniLcs.Cap.FeedPoller`	Atom feed polling GenServer. Schedules periodic fetches for each configured feed URL.

Both start automatically and are supervised with `:one_for_one` strategy alongside the existing CBC and E-SMLC processes.

Troubleshooting

No Cells Matched by Polygon

Symptoms: Alert is accepted but `matched_cells` is 0 and TAC/LAC lists are empty.

Possible causes:

- Cell database has no cells with geographic positions inside the warning polygon
- Cells exist in the polygon area but lack `tac` and `lac` fields
- Polygon coordinates in the CAP XML are in the wrong order or format

Resolution:

1. Verify cells in the warning area have `tac` and/or `lac` values set via `GET /api/cells`
2. Check that cell positions (latitude/longitude) are correct
3. Try enabling `coverage_aware: true` if cells are near polygon edges
4. Validate the CAP polygon coordinates against a map

Alert Stuck in Pending

Symptoms: Alert appears as pending but operator cannot approve.

Possible causes:

- `require_approval` is `true` and no operator has approved the alert
- LiveView page not connected (check browser WebSocket connection)

Resolution:

1. Navigate to `/cap` in the Control Panel and click Approve
2. Alternatively, use the REST API: `PUT /api/cap/<id>` with `{"action": "approve", "operator": "operator1"}`

Feed Poller Not Picking Up Alerts

Symptoms: Configured feed URL has new alerts but they don't appear in OmniLCS.

Possible causes:

- Feed URL is incorrect or unreachable

- Feed format is not standard Atom with embedded CAP
- TLS certificate issues
- Alert entries don't contain `<content>` with embedded `<alert>` XML

Resolution:

1. Check logs for `CAP FeedPoller: Failed to fetch` messages
2. Verify the feed URL returns valid Atom XML
3. Confirm the feed contains `<entry>` elements with `<content>` containing `<alert>` XML
4. For TLS issues, check that the feed URL is accessible from the OmniLCS server

Broadcast Sent but Not Received by UEs

Symptoms: Alert status shows "Sent" but no UEs receive the broadcast message.

Possible causes:

- TAC/LAC values don't match the actual network configuration
- MMEs or BSCs rejected the broadcast (check CBC 4G / CBC pages for response details)
- Message encoding issues

Resolution:

1. Verify TAC values match the MME's tracking area configuration
2. Check the CBC 4G page for Write-Replace-Warning-Response status
3. Check the CBC page for WRITE-REPLACE COMPLETE/FAILURE messages
4. Review broadcast results in the alert history (expand the alert row)

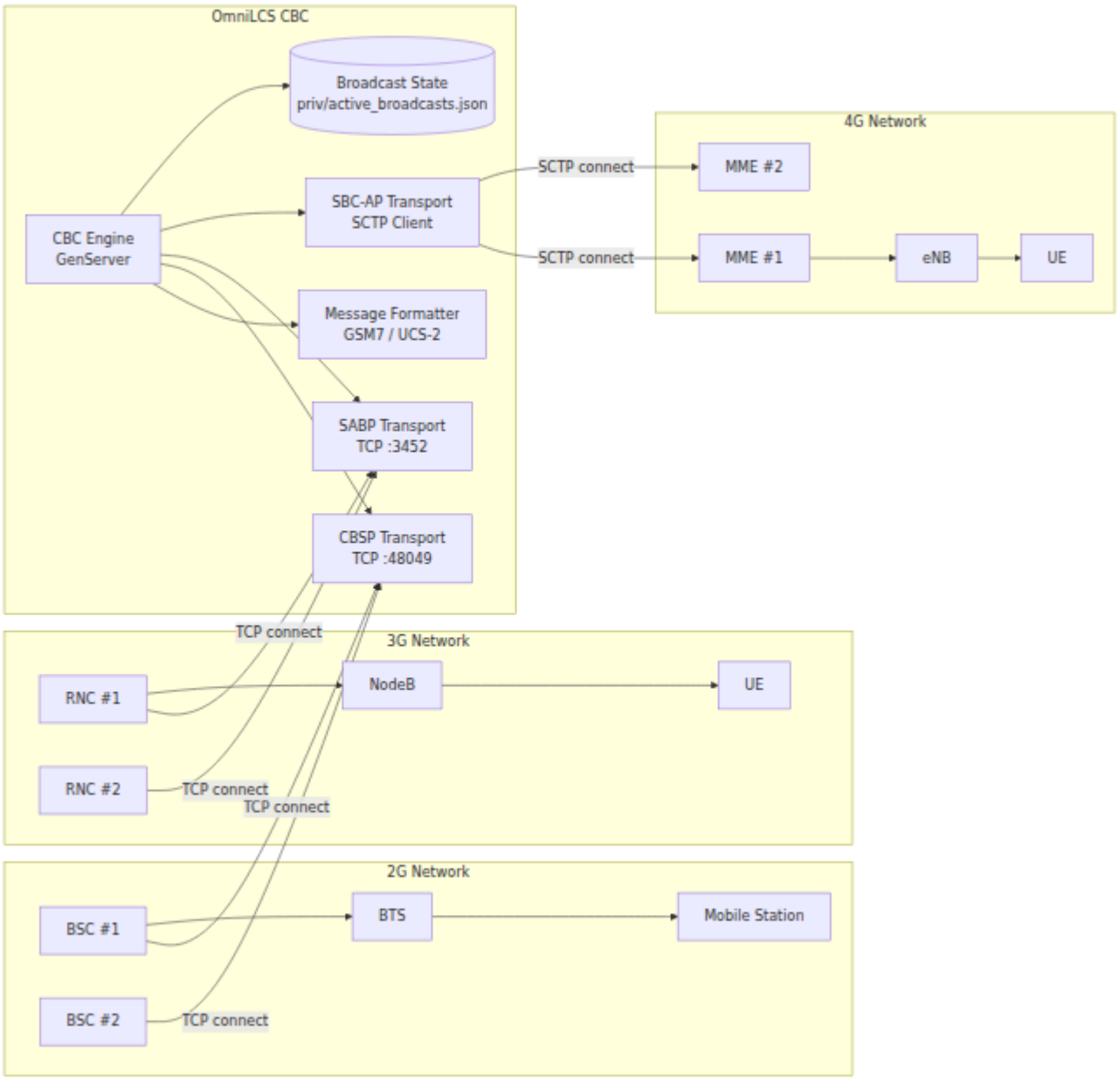
3GPP and Standards References

Specification	Title
OASIS CAP v1.2	Common Alerting Protocol Version 1.2
RFC 4287	The Atom Syndication Format
TS 29.168	Cell Broadcast Centre interfaces with the Evolved Packet Core (SBC-AP)
TS 48.049	Cell Broadcast Centre Protocol (CBSP)
TS 23.041	Technical realization of Cell Broadcast Service (CBS)
TS 23.038	Alphabets and language-specific information

Cell Broadcast Operations Guide

OmniLCS implements a Cell Broadcast Centre (CBC) supporting 2G networks via CBSP, 3G UTRAN networks via SABP, and 4G LTE networks via SBC-AP. The CBC can send, update, and stop broadcast messages across all connected radio access network elements simultaneously.

Architecture



2G CBSP Operations

Connection Model

CBSP uses TCP transport per 3GPP TS 48.049. The CBC listens on TCP port **48049** (IANA registered) and BSCs establish inbound connections.

- Each accepted connection spawns a connection handler under a DynamicSupervisor

- Connections are tracked in the `:cbsp_connections` ETS table
- Connection state is broadcast via PubSub to the LiveView UI

Keep-Alive

The CBC implements proactive keep-alive:

Parameter	Default	Description
Keep-alive interval	30 seconds	How often the CBC sends KEEP-ALIVE to each peer
Keep-alive timeout	10 seconds	Time to wait for KEEP-ALIVE COMPLETE before marking unhealthy

The CBC both sends proactive KEEP-ALIVE messages and responds to KEEP-ALIVE messages initiated by BSCs with KEEP-ALIVE COMPLETE.

CBSP Message Types

Message	Direction	Code	Description
WRITE-REPLACE	CBC -> BSC	0x01	Send or update a broadcast message
WRITE-REPLACE COMPLETE	BSC -> CBC	0x02	Broadcast accepted by BSC
WRITE-REPLACE FAILURE	BSC -> CBC	0x03	Broadcast rejected by BSC
KILL	CBC -> BSC	0x04	Stop a broadcast
KILL COMPLETE	BSC -> CBC	0x05	Broadcast stopped
KILL FAILURE	BSC -> CBC	0x06	Failed to stop broadcast
LOAD-QUERY	CBC -> BSC	0x07	Query radio resource loading
LOAD-QUERY COMPLETE	BSC -> CBC	0x08	Load information returned
LOAD-QUERY FAILURE	BSC -> CBC	0x09	Load query failed
STATUS-QUERY	CBC -> BSC	0x0A	Query broadcast delivery status
STATUS-QUERY COMPLETE	BSC -> CBC	0x0B	Status information returned

Message	Direction	Code	Description
STATUS-QUERY FAILURE	BSC -> CBC	0x0C	Status query failed
RESET	CBC -> BSC	0x10	Reset broadcast state on BSC
RESET COMPLETE	BSC -> CBC	0x11	Reset acknowledged
RESET FAILURE	BSC -> CBC	0x12	Reset failed
RESTART	BSC -> CBC	0x13	BSC restarted (informational)
FAILURE	BSC -> CBC	0x14	BSC failure indication
ERROR INDICATION	Either	0x15	Protocol error
KEEP-ALIVE	Either	0x16	Connection supervision
KEEP-ALIVE COMPLETE	Either	0x17	Keep-alive response

CBSP Message Framing

Every CBSP message on the wire is:

```
+----------+----------+------+
| Length   | Type     | IEs  |
| (3 bytes)| (1 byte) | ...  |
+----------+----------+------+
```

The 3-byte length field covers the type byte plus all IEs (excludes the 3 length bytes themselves).

Cell List Formats

The Cell List IE identifies which cells a broadcast targets. Supported formats:

Discriminator	Value	Format	Description
CGI	0x00	MCC+MNC+LAC+CI	Full Cell Global Identity
LAC+CI	0x01	PLMN+LAC+CI	Location Area Code + Cell Identity
CI	0x02	CI only	Cell Identity only
LAI	0x04	MCC+MNC+LAC	Location Area Identity
LAC	0x05	LAC only	Location Area Code
All in BSC	0x06	(no cells)	All cells managed by the BSC

Channel Indicator

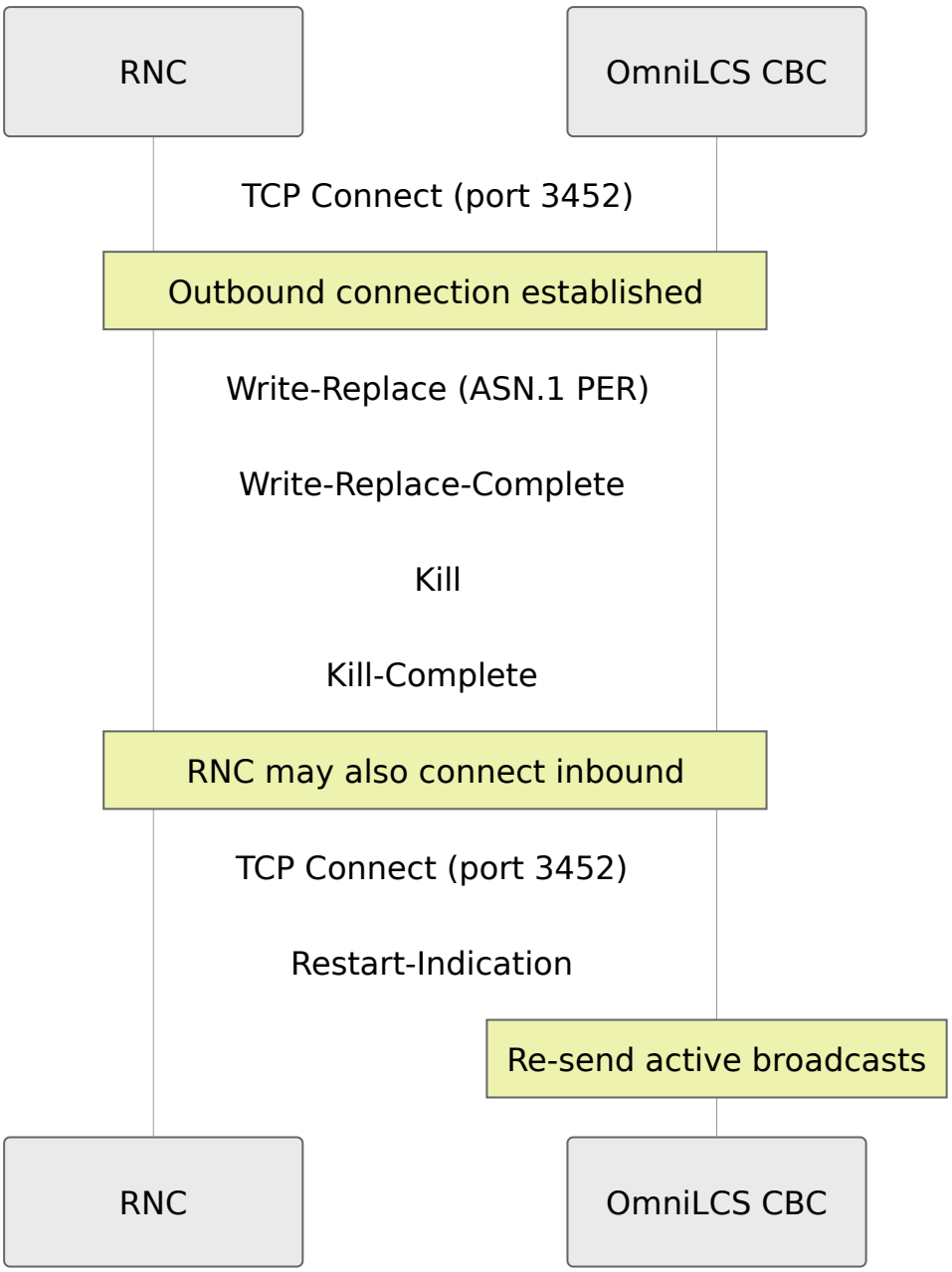
Value	Name	Description
0	Basic CBCH	Primary Cell Broadcast Channel
1	Extended CBCH	Extended Cell Broadcast Channel (higher capacity)

3G SABP Operations

The Service Area Broadcast Protocol (SABP) provides the Iu-BC interface between the CBC and RNCs in a 3G UTRAN network, per 3GPP TS 25.419. SABP uses ASN.1 PER aligned encoding over TCP.

Connection Model

SABP uses TCP transport per TS 25.414 §7.1.3.3. The IANA-registered port for SABP is **3452**. Per TS 25.419 §5, the CBC initiates connections for normal operations (Write-Replace, Kill, etc.) and the RNC initiates connections only for Failure/Restart Indications. Both sides use destination port 3452 when establishing new connections.



- Each accepted TCP connection spawns a connection handler under the SABP connection DynamicSupervisor
- Connections are tracked in the `:sabp_connections` ETS table
- Connection state changes are broadcast via PubSub to the LiveView UI
- TCP keepalive socket option is enabled since SABP has no protocol-level keep-alive mechanism

SABP Message Framing

SABP messages are framed over TCP with a 4-byte big-endian length prefix:

```
+-------------------+--------------------------+
| Length (4 bytes)   | ASN.1 PER Encoded PDU     |
| big-endian uint32  | ...                       |
+-------------------+--------------------------+
```

The length field specifies the number of bytes in the ASN.1 PER encoded payload that follows. Multiple messages may arrive in a single TCP segment; the connection handler buffers and reassembles as needed.

SABP Procedures

Procedure	Code	Class	Direction	Description
Write-Replace	0	Class 1	CBC -> RNC	Send or update a broadcast message
Kill	1	Class 1	CBC -> RNC	Stop an active broadcast
Load-Status-Enquiry	2	Class 1	CBC -> RNC	Query radio resource loading
Message-Status-Query	3	Class 1	CBC -> RNC	Query broadcast delivery status
Reset	4	Class 1	CBC -> RNC	Reset broadcast state on RNC
Restart-Indication	5	Class 2	RNC -> CBC	RNC restarted, re-send active broadcasts
Failure-Indication	6	Class 2	RNC -> CBC	RNC failure indication
Error-Indication	7	Class 2	Either	Report a protocol error

SABP Information Elements

The SABP protocol uses ASN.1-defined Information Elements (IEs). Key IEs used in Write-Replace:

IE	ID	Criticality	Type	Description
Broadcast-Message-Content	0	Reject	Binary	Encoded CBS message content
Category	1	Ignore	Enum	Message priority (high, normal, background, default)
Cause	2	Ignore	Integer	Cause value for failure/error indications
Data-Coding-Scheme	4	Ignore	8-bit	How message content is encoded (e.g., 0x0F for GSM 7-bit)
Message-Identifier	6	Reject	16-bit	CB message identifier
New-Serial-Number	7	Reject	16-bit	Serial number for the new/updated broadcast
Number-of-Broadcasts-Requested	9	Reject	Integer	Total broadcast count (0..65535)
Old-Serial-Number	10	Ignore	16-bit	Serial number of message being replaced (optional)
Repetition-Period	13	Reject	Integer	Seconds between repetitions (1..4096)
Service-Areas-List	15	Reject	List	Target service areas (list of SAIs)

Service Area Identifier (SAI)

SABP targets broadcasts to specific Service Areas identified by SAIs. Each SAI consists of:

Field	Size	Description
PLMN Identity	3 bytes	BCD-encoded MCC+MNC per 3GPP TS 24.008
LAC	2 bytes	Location Area Code
SAC	2 bytes	Service Area Code

Broadcast Category

The optional Category IE sets the priority of a broadcast message:

Category	Description
high-priority	Highest priority, pre-empts other broadcasts
normal-priority	Standard priority
background-priority	Low priority, does not pre-empt
default-priority	Default when no category is specified

Restart-Indication and Failure-Indication Handling

When the CBC receives a **Restart-Indication** from an RNC, the RNC has restarted and lost its broadcast state. The CBC should re-send all active broadcasts to that RNC.

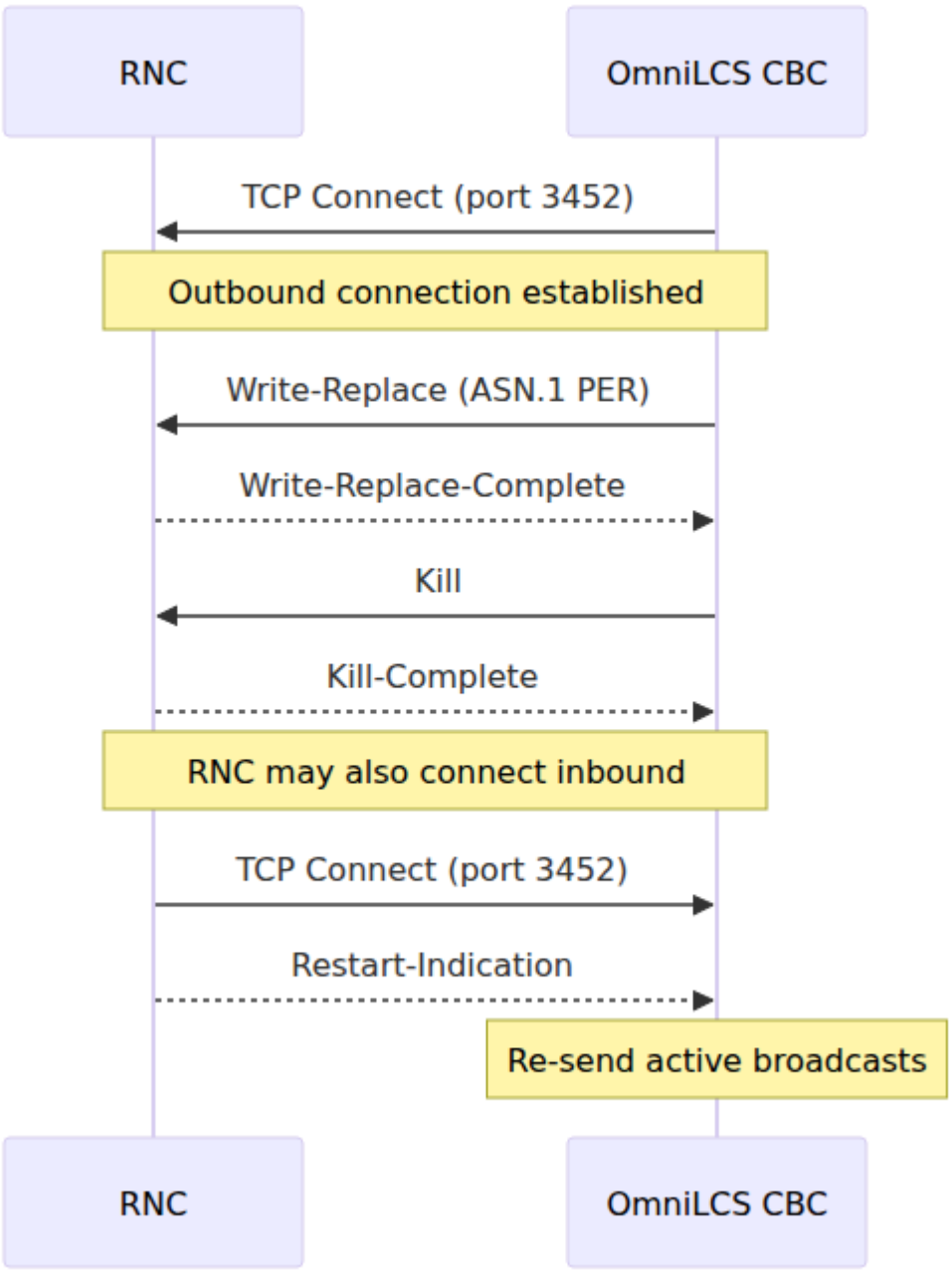
When the CBC receives a **Failure-Indication**, the RNC is reporting that certain service areas have lost broadcast capability. This is logged and emitted as a

telemetry event.

4G SBC-AP Operations

Connection Model

Per 3GPP TS 29.168 Section 4.3, the **CBC initiates SCTP associations** to each MME. The MME listens on port **29168** (IANA registered). The SCTP Payload Protocol Identifier (PPID) for SBC-AP is **24**.



The transport implements automatic reconnection with exponential backoff:

Parameter	Value	Description
Initial reconnect delay	5 seconds	First retry delay after connection failure
Maximum reconnect delay	60 seconds	Maximum backoff ceiling
SCTP heartbeat interval	10 seconds	Peer address heartbeat
SCTP path max retransmissions	7	Before declaring path failure
SCTP RTO max	30 seconds	Maximum retransmission timeout
SCTP RTO min	1 second	Minimum retransmission timeout

SBC-AP Procedures

Procedure	Code	Class	Direction	Description
Write-Replace-Warning	0	Class 1	CBC -> MME	Send or update a warning message
Stop-Warning	1	Class 1	CBC -> MME	Stop an active warning
Error-Indication	2	Class 2	Either	Report a protocol error
PWS-Restart-Indication	3	Class 2	MME -> CBC	MME restarted, re-send active warnings
PWS-Failure-Indication	4	Class 2	MME -> CBC	eNB failure, cells lost broadcast capability

Write-Replace-Warning-Request IEs

IE	ID	Criticality	Type	Description
Message-Identifier	5	Reject	16-bit	CB message identifier (e.g., 0x1112 for CMAS)
Serial-Number	11	Reject	16-bit	Serial number with geo scope, msg code, update number
List-of-TAIs	14	Reject	List	Tracking area identities for the broadcast
Warning-Area-List	15	Ignore	Choice	Target area (TAI list, cell list, or emergency area)
Repetition-Period	10	Reject	Integer	Seconds between repetitions (0..4096)
Number-of-Broadcasts-Requested	7	Reject	Integer	Total broadcast count (0..65535)
Warning-Type	18	Ignore	2 bytes	ETWS warning type + activation flags
Data-Coding-Scheme	3	Ignore	8-bit	How message content is encoded
Warning-Message-Content	16	Ignore	Binary	Encoded CBS pages
Omc-Id	19	Ignore	Binary	Operations centre identifier

IE	ID	Criticality	Type	Description
Concurrent-Warning-Message-Indicator	20	Reject	Boolean	Allow concurrent warnings

Stop-Warning-Request IEs

IE	ID	Criticality	Description
Message-Identifier	5	Reject	CB message identifier to stop
Serial-Number	11	Reject	Serial number of the broadcast to stop
List-of-TAIs	14	Reject	(Optional) Limit stop to specific TAIs
Warning-Area-List	15	Ignore	(Optional) Limit stop to specific areas

PWS-Restart-Indication Handling

When the CBC receives a PWS-Restart-Indication from an MME, it re-sends all active broadcasts (those in `:sent` or `:acknowledged` status) to that MME. This ensures broadcast continuity after MME restarts.

PWS-Failure-Indication Handling

When the CBC receives a PWS-Failure-Indication, it marks affected broadcasts with a `:pws_failure` response for the reporting MME association. This is logged as a warning.

Message Formatting

GSM 7-bit Encoding

The default encoding for CBS messages. Each CBS page holds up to **93 septets** packed into **82 bytes**.

The GSM 7-bit default alphabet maps standard Latin characters plus Greek letters and special symbols. Characters not in the basic alphabet use the extension table (escape sequence 0x1B + extension code), consuming two septets.

The Warning-Message-Content binary structure:

```
+----------+-----------------------------------------+
| Num Pages| Page 1 (82 bytes packed + 1 byte length) |
| (1 byte) | Page 2 ...                               |
+----------+-----------------------------------------+
```

UCS-2 Encoding

For non-Latin scripts (CJK, Arabic, Thai, etc.). Each CBS page holds up to **40 UCS-2 characters** (80 bytes of UTF-16BE data), padded to 82 bytes.

Data Coding Scheme (DCS)

Encoding	DCS Value	Description
GSM 7-bit	0x0F	GSM 7-bit default alphabet, language unspecified
UCS-2	0x48	UCS-2 (UTF-16BE), language unspecified

The DCS can be set explicitly per broadcast or is auto-derived from the encoding parameter.

Serial Number Structure

Per 3GPP TS 23.041 Section 9.4.1.2.1, the 16-bit serial number has internal structure:

```
+----+------------+--------+
| GS | Msg Code   | Update |
| 2b | 10 bits    | 4 bits |
+----+------------+--------+
```

Field	Bits	Description
Geographical Scope (GS)	15-14	0 = Cell-wide immediate, 1 = PLMN-wide, 2 = LA/TA-wide, 3 = Cell-wide
Message Code	13-4	Identifies the broadcast within its scope (0..1023)
Update Number	3-0	Incremented for message updates (0..15)

The `MessageFormatter.build_serial_number/3` helper constructs a serial number from these components.

Warning Types (ETWS)

The Warning-Type IE is 2 bytes per 3GPP TS 23.041 Section 9.3.24:

Warning Type	Octet 1 Value	Description
Earthquake	0x00	Earthquake warning
Tsunami	0x01	Tsunami warning
Earthquake + Tsunami	0x02	Combined earthquake and tsunami
Test	0x03	Test warning
Other	0x04	Other emergency

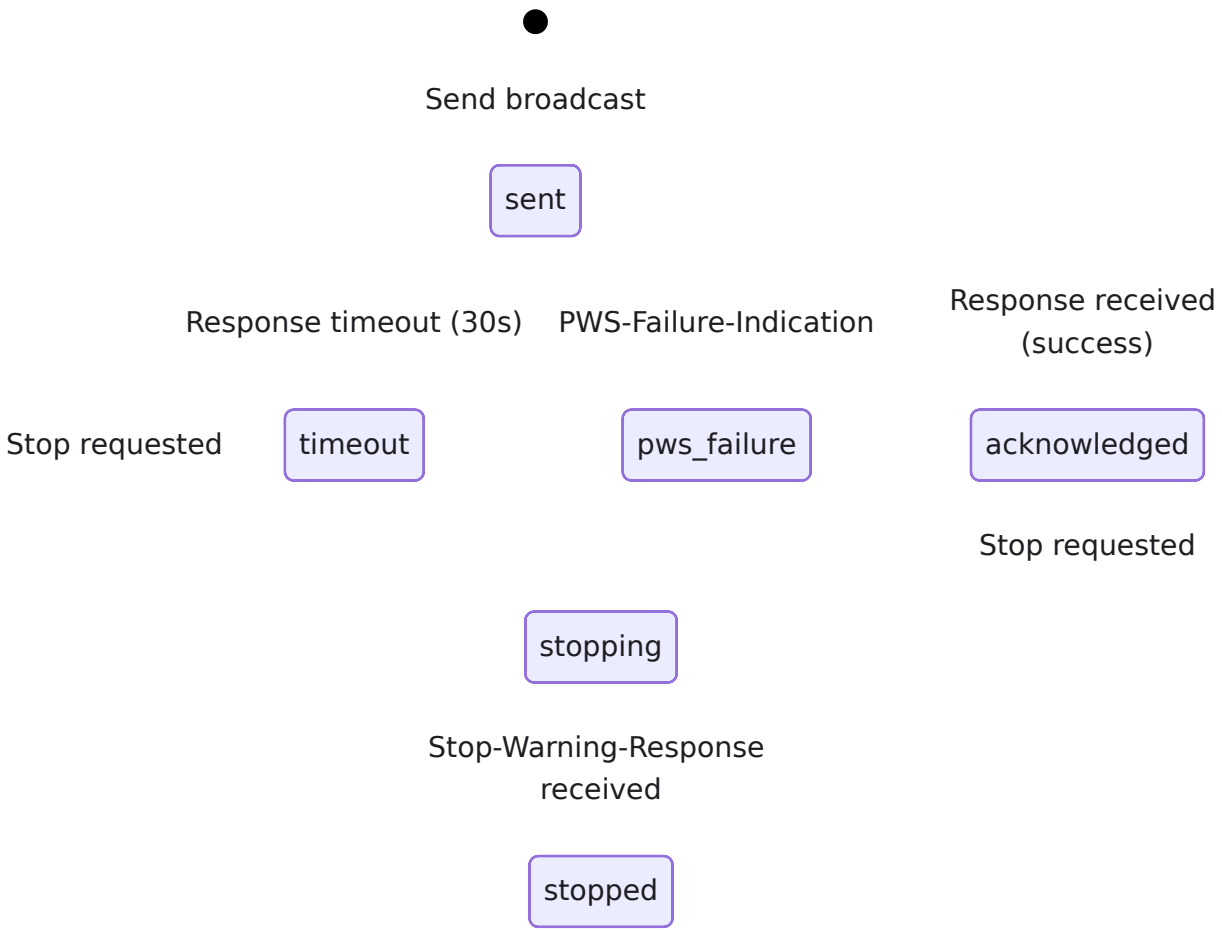
Octet 2 activation flags (default 0xC0):

Bit	Value	Description
Bit 8	0x80	Emergency User Alert (activate alert tone/vibration)
Bit 7	0x40	Popup (automatically display message on screen)

Both bits are set by default (0xC0) to ensure maximum visibility.

Broadcast State Management

State Lifecycle



Persistence

Active broadcasts for 2G, 3G, and 4G are persisted to `priv/active_broadcasts.json` as JSON. On startup, the Engine reloads this file to restore broadcast state. This allows the CBC to re-send active warnings after application restart. The persistence file stores three separate broadcast maps: `active_broadcasts_4g`, `active_broadcasts_3g`, and `active_broadcasts_2g3g`.

Response Tracking

For 4G broadcasts, each `send_broadcast_4g` starts a 30-second response timeout timer. Responses from MMEs are correlated by Message-Identifier and

Serial-Number. The timer is cancelled when all expected MME peers have responded. Timed-out peers are logged with a `:timeout` cause.

Active Broadcasts View

The Active Broadcasts page provides a unified view of all in-flight broadcasts across 2G, 3G, and 4G networks.

Broadcast History

The Engine maintains a rolling history of the last 100 4G broadcasts for the LiveView UI and API queries.

Troubleshooting

No MMEs Connected (SBC-AP)

1. Verify `mme_peers` configuration contains correct IP addresses and ports
2. Check that the `local_ip` is reachable from the MME network
3. Look for SCTP connection errors in the log: `SBC-AP: Failed to connect to MME`

4. Verify SCTP is not blocked by firewalls (protocol 132)
5. Confirm MME is listening on port 29168

No BSCs Connected (CBSP)

1. Verify `listen_ip` and `listen_port` configuration under `:cbsp`
2. Check that TCP port 48049 is not blocked by firewalls
3. Look for accept errors: `CBSP accept failed`
4. Confirm BSC is configured with the CBC IP address and port

No RNCs Connected (SABP)

1. Verify `listen_ip` and `listen_port` configuration under `:sabp`
2. Check that TCP port 3452 is not blocked by firewalls
3. Look for accept errors: `SABP accept failed`
4. Confirm RNC is configured with the CBC IP address and port
5. Check logs for `SABP connection closed by peer` or `SABP TCP error` messages

Broadcast Not Delivered

1. Check the active broadcasts table for response status
2. Look for WRITE-REPLACE FAILURE or KILL FAILURE messages in the CBSP/SABP message log
3. Verify cell list format matches what the BSC/RNC/MME expects
4. For 4G, check that TAC values are valid for the target MME
5. For 3G, check that Service Area Identifiers (SAIs) are valid for the target RNC
6. Check the broadcast history for timeout entries

Encoding Issues

1. For non-Latin text, ensure `:ucs2` encoding is selected
2. Verify the DCS matches the encoding (0x0F for GSM 7-bit, 0x48 for UCS-2)

3. GSM 7-bit can only represent the GSM default alphabet; unsupported characters are replaced with `?`

3GPP References

Specification	Title
TS 25.419	UTRAN Iu-BC interface: Service Area Broadcast Protocol (SABP)
TS 29.168	Cell Broadcast Centre interfaces with the Evolved Packet Core (SBC-AP)
TS 48.049	Cell Broadcast Centre Protocol (CBSP)
TS 23.041	Technical realization of Cell Broadcast Service (CBS)
TS 23.038	Alphabets and language-specific information (DCS, GSM 7-bit)
TS 24.008	Mobile radio interface layer 3 specification (PLMN identity encoding)

OmniLCS Configuration Reference

All configuration is defined in `config/config.exs`. This document covers every configuration section and parameter.

Control Panel (LiveView UI)

The Control Panel provides the web-based management interface served over HTTPS on port 443.

```
config :control_panel,
  parent_application: :omnilcs,
  parent_application_version: "1.0.0",
  parent_application_readable_name: "OmniLCS",
  home_page: ControlPanelWeb.ApplicationLive,
  use_builtin_pages: [...],
  use_additional_pages: [...],
  page_order: ["/dashboard", "/location", "/cells", "/diameter",
               "/cbc", "/cbc4g", "/application", "/configuration",
"/log"],
  licensee_name: "Omnitouch"
```

Parameter	Type	Description
`parent_application`	atom	OTP application name
`parent_application_version`	string	Version displayed in the UI
`parent_application_readable_name`	string	Human-readable product name
`home_page`	module	Default landing page module
`use_builtin_pages`	list	Built-in pages (Application, Configuration, Log)
`use_additional_pages`	list	Custom OmniLCS pages (Dashboard, Location, Cells, Diameter, CBC 2G, CBC 3G, CBC 4G, CAP Alerts)
`page_order`	list	Tab ordering in the navigation bar
`licensee_name`	string	Licensee name displayed in the footer

Control Panel Endpoint

```
config :control_panel, ControlPanelWeb.Endpoint,
  server: true,
  https: [
    port: 443,
    keyfile: "priv/cert/omnitouch.pem",
    certfile: "priv/cert/omnitouch.crt"
  ],
  secret_key_base: "...",
  check_origin: false,
  pubsub_server: ControlPanel.PubSub,
  live_view: [signing_salt: "LcsLvSlt"]
```

Parameter	Type	Default	Des
`server`	boolean	`true`	Enable server
`https.port`	integer	`443`	HTTPS port
`https.keyfile`	string	`"priv/cert/omnitouch.pem"`	Path to private
`https.certfile`	string	`"priv/cert/omnitouch.crt"`	Path to certific
`secret_key_base`	string	--	Phoen encryp (gener `mix phx.ge`
`check_origin`	boolean	`false`	Wheth validat WebSo origins
`pubsub_server`	atom	`ControlPanel.PubSub`	PubSu for Liv update
`live_view.signing_salt`	string	`"LcsLvSlt"`	LiveVie signing

REST API

The REST API is served over HTTPS on port 8443 via the api_ex framework.

```
config :api_ex,
  api: %{
    port: 8443,
    listen_ip: "0.0.0.0",
    product_name: "OmniLCS",
    title: "API - OmniLCS",
    hostname: "localhost",
    enable_tls: true,
    tls_cert_path: "priv/cert/omnitouch.crt",
    tls_key_path: "priv/cert/omnitouch.pem",
    routes: [...]
  }
```

Parameter	Type	Default	Description
`port`	integer	`8443`	API listen port
`listen_ip`	string	`"0.0.0.0"`	API bind address
`product_name`	string	`"OmniLCS"`	Product name in OpenAPI spec
`title`	string	`"API - OmniLCS"`	API title in Swagger UI
`hostname`	string	`"localhost"`	Hostname for OpenAPI spec URLs
`enable_tls`	boolean	`true`	Enable HTTPS
`tls_cert_path`	string	`"priv/cert/omnitouch.crt"`	Path to TLS certificate
`tls_key_path`	string	`"priv/cert/omnitouch.pem"`	Path to TLS private key
`routes`	list	--	List of route definitions (path, module, actions)

Registered API Routes

Path	Controller	Actions
`/status`	`OmniLcs.Api.StatusController`	`index`
`/location`	`OmniLcs.Api.LocationController`	`index`, `create`, `show`
`/cells`	`OmniLcs.Api.CellController`	`index`, `create`, `show`, `update`, `delete`
`/cap`	`OmniLcs.Api.CapController`	`index`, `create`, `show`, `update`

E-SMLC Settings

General E-SMLC configuration.

```
config :omnilcs,
  esmlc_name: "OmniLCS",
  cell_database_path: "priv/cells.json"
```

Parameter	Type	Default	Description
`esmlc_name`	string	`"OmniLCS"`	E-SMLC instance name (used in status API)
`cell_database_path`	string	`"priv/cells.json"`	Path to cell database JSON file for import

CBSP (2G Cell Broadcast)

Configuration for the CBSP TCP listener. BSCs connect inbound to the CBC on this port.

```
config :omnilcs, :cbsp,
  listen_ip: "0.0.0.0",
  listen_port: 48049
```

Parameter	Type	Default	Description
`listen_ip`	string	`"0.0.0.0"`	IP address to bind the CBSP TCP listener
`listen_port`	integer	`48049`	TCP port for CBSP connections (IANA registered port for CBSP)

SABP (3G Cell Broadcast)

Configuration for the SABP TCP listener. RNCs connect inbound to the CBC on this port via the Iu-BC interface per 3GPP TS 25.419.

```
config :omnilcs, :sabp,
  listen_ip: "0.0.0.0",
  listen_port: 3452
```

Parameter	Type	Default	Description
`listen_ip`	string	`"0.0.0.0"`	IP address to bind the SABP TCP listener
`listen_port`	integer	`3452`	TCP port for SABP connections from RNCs (IANA-registered, per TS 25.414 §7.1.3.3)

SBC-AP (4G Cell Broadcast)

Configuration for SBC-AP SCTP connections. Per 3GPP TS 29.168, the CBC initiates SCTP associations to each MME peer.

```
config :omnilcs, :sbcap,
  local_ip: "10.5.198.200",
  mme_peers: [
    %{host: "mme01", ip: "10.179.2.100", port: 29168},
    %{host: "mme02", ip: "10.179.2.101", port: 29168}
  ]
```

Parameter	Type	Default	Description
`local_ip`	string	`"0.0.0.0"`	Local IP address to bind the SCTP socket
`mme_peers`	list	`[]`	List of MME peer configurations

MME Peer Configuration

Each entry in `mme_peers` is a map with the following fields:

Field	Type	Required	Default	Description
`host`	string	No	value of `ip`	Human-readable MME hostname (for logging and UI)
`ip`	string	Yes	--	MME IP address
`port`	integer	No	`29168`	MME SBC-AP SCTP port (IANA registered)

The SBC-AP transport uses SCTP Payload Protocol Identifier (PPID) 24. On connection failure, it retries with exponential backoff starting at 5 seconds up to a maximum of 60 seconds.

InfluxDB

Cell position data is periodically synchronized from an InfluxDB instance.

```
config :omnilcs, OmniLcs.InfluxDb,
  database: "nokia-monitor",
  host: "172.19.3.68",
  port: 8086,
  auth: [method: :basic, username: "monitor", password: "..."],
  http_opts: [recv_timeout: 30_000],
  pool: [max_overflow: 10, size: 5]
```

Parameter	Type	Default	Description
`database`	string	--	InfluxDB database name
`host`	string	--	InfluxDB server hostname or IP
`port`	integer	`8086`	InfluxDB HTTP API port
`auth.method`	atom	`:basic`	Authentication method
`auth.username`	string	--	InfluxDB username
`auth.password`	string	--	InfluxDB password
`http_opts.recv_timeout`	integer	`30000`	HTTP receive timeout in milliseconds
`pool.size`	integer	`5`	Connection pool size
`pool.max_overflow`	integer	`10`	Maximum pool overflow connections

Cell sync runs automatically every 5 minutes with an initial delay of 10 seconds after startup. It can also be triggered manually via the REST API or the Control Panel.

SLs Interface (LCS-AP over SCTP)

The SLs interface connects the E-SMLC to the MME using LCS-AP over SCTP per 3GPP TS 29.171. OmniLCS initiates SCTP associations to each configured MME peer on port 9082 with PPID 29.

```
config :omnilcs, :sls,
  local_ip: "10.5.198.200",
  mme_peers: [
    %{host: "mme01", ip: "10.179.1.15", port: 9082}
  ]
```

Parameter	Type	Default	Description
`local_ip`	string	`"0.0.0.0"`	Local IP address to bind the SCTP socket
`mme_peers`	list	`[]`	List of MME peer configurations

MME Peer Configuration (SLs)

Each entry in `mme_peers` is a map with the following fields:

Field	Type	Required	Default	Description
`host`	string	No	value of `ip`	Human-readable MME hostname (for logging and UI)
`ip`	string	Yes	--	MME IP address
`port`	integer	No	`9082`	MME LCS-AP SCTP port (IANA registered for LCS-AP)

The SLs transport uses SCTP Payload Protocol Identifier (PPID) 29. On connection failure, it retries with exponential backoff starting at 5 seconds up to a maximum of 60 seconds.

SCTP tuning parameters:

Parameter	Value	Description
Heartbeat interval	10 seconds	Peer address heartbeat
Path max retransmissions	7	Before declaring path failure
RTO max	30 seconds	Maximum retransmission timeout
RTO min	1 second	Minimum retransmission timeout
SACK delay	200 ms	Selective acknowledgment delay

Diameter (SLg and Other Interfaces)

The Diameter configuration controls the SLg interface between OmniLCS (acting as GMLC) and the MME, routed through a DRA. This is separate from the SLs interface which uses native LCS-AP over SCTP.

```
config :diameter_ex,
  diameter: %{
    service_name: :omnitouch_esmlc,
    listen_ip: "10.5.198.200",
    listen_port: 3868,
    host: "amanaki",
    realm: "epc.mnc380.mcc313.3gppnetwork.org",
    product_name: "OmniLCS",
    request_timeout: 5000,
    peer_selection_algorithm: :random,
    allow_undefined_peers_to_connect: true,
    log_unauthorized_peer_connection_attempts: true,
    control_module: OmniLcs.Control.Diameter,
    vendor_id: 10415,
    supported_vendor_ids: [5535, 10415],
    applications: [...],
    peers: [...]
  }
```

Service Parameters

Parameter	Type	Defaul
`service_name`	atom	`:omnitouch_esml`
`listen_ip`	string	--
`listen_port`	integer	`3868`
`host`	string	--
`realm`	string	--
`product_name`	string	`"OmniLCS"`
`request_timeout`	integer	`5000`
`peer_selection_algorithm`	atom	`:random`

Parameter	Type	Defaul
`allow_undefined_peers_to_connect`	boolean	`true`
`log_unauthorized_peer_connection_attempts`	boolean	`true`
`control_module`	module	`OmniLcs.Control`
`processor_module`	module	`DiameterEx.Proc`
`vendor_id`	integer	`10415`
`supported_vendor_ids`	list	`[5535, 10415]`

Diameter Applications

```
applications: [
  %{
    application_name: :slg,
    application_dictionary: :diameter_gen_3gpp_slg,
    vendor_specific_application_ids: [
      %{vendor_id: 10415, auth_application_id: 16_777_264,
acct_application_id: nil}
    ]
  }
]
```

Field	Type	Description
`application_name`	atom	Application identifier (`:slg` for GMLC-to-MME interface)
`application_dictionary`	atom	Erlang Diameter dictionary module
`vendor_specific_application_ids`	list	Vendor-Specific-Application-Id AVP values

The SLg application uses Application ID **16777264** with 3GPP Vendor-Id **10415**. Note that the E-SMLC-to-MME interface (SLs) uses native LCS-AP over SCTP, not Diameter.

Diameter Peers

```
peers: [
  %{
    host: "omni-nick2-dra01.epc.mnc380.mcc313.3gppnetwork.org",
    realm: "epc.mnc380.mcc313.3gppnetwork.org",
    ip: "10.179.2.233",
    port: 3868,
    tls: false,
    transport: :diameter_sctp,
    initiate_connection: true
  }
]
```

Field	Type	Default	Description
`host`	string	--	Peer Diameter host identity (FQDN)
`realm`	string	--	Peer Diameter realm
`ip`	string	--	Peer IP address
`port`	integer	`3868`	Peer Diameter port
`tls`	boolean	`false`	Enable TLS for the peer connection
`transport`	atom	`:diameter_sctp`	Transport protocol (`:diameter_sctp` or `:diameter_tcp`)
`initiate_connection`	boolean	`true`	Whether OmniLCS initiates the connection to this peer

GMLC / Le Interface

Configuration for the Gateway Mobile Location Centre and Le interface to external LCS clients. See the GMLC & Le Interface Operations Guide for full details on request flows, deferred sessions, and InfluxDB logging.

```
config :omnilcs, :gmlc,
  enabled: true,
  allow_unknown_clients: false,
  authorized_clients: [
    %{
      name: "psap-01",
      type: :emergency_services,
      allowed_methods: [:cell, :ecid, :gnss, :otdoa],
      rate_limit: 100,
      description: "Primary PSAP"
    }
  ],
  allow_deferred: true,
  max_periodic_sessions: 100,
  max_triggered_sessions: 50,
  default_periodic_poll_interval_ms: 60_000,
  default_triggered_poll_interval_ms: 30_000,
  influx_logging: true
```

Parameter	Type	Default	Description
`enabled`	boolean	`false`	Enable the GMLC Le interface handler
`allow_unknown_clients`	boolean	`false`	Accept requests from unlisted clients
`authorized_clients`	list	`[]`	Authorized LCS client definitions (see GMLC guide)
`allow_deferred`	boolean	`true`	Accept periodic and triggered location requests
`max_periodic_sessions`	integer	`100`	Maximum concurrent periodic sessions
`max_triggered_sessions`	integer	`50`	Maximum concurrent triggered sessions
`default_periodic_poll_interval_ms`	integer	`60000`	Default periodic fix

Parameter	Type	Default	Description
			interval (ms)
`default_triggered_poll_interval_ms`	integer	`30000`	Default geo-fence poll interval (ms)
`influx_logging`	boolean	`true`	Write all GMLC location fixes to InfluxDB

2G/3G Location (MAP / Lg)

GMLC location for subscribers on GERAN (2G) or UTRAN (3G), reached over the **Lg** interface using MAP on the `omniss7` SS7/SIGTRAN stack. Disabled by default; see 2G/3G location over MAP/Lg.

```
# OmniLCS MAP/Lg client (OmniLcs.Control.MapClient)
config :omnilcs, :map,
  enabled: true,
  gmlc_number: "61400000000",   # this GMLC's own ISDN number
(E.164)
  hlr_gt: "61400000001",        # HLR global title for
SendRoutingInfoForLCS
  lcs_client_type: :emergencyServices

# Run the omniss7 SS7 stack as a library (no own web UI / SCTP /
license client)
config :omniss7, headless: true

# SS7 transport for the MAP client — the M3UA association to the
STP.
# Only needed when :omnilcs, :map is enabled.
config :omniss7, :map_client_enabled, true

config :omniss7, :map_client_m3ua, %{
  local_ip: {10, 0, 0, 100},
  local_port: 2905,
  remote_ip: {10, 0, 0, 1},     # STP / SGP
  remote_port: 2905,
  routing_context: 1
}
```

Parameter	Type	Default	Description
`:omnilcs, :map enabled`	boolean	`false`	Enable the 2G/3G MAP/Lg location path. When `false`, `rat: "2g"`/`"3g"` requests return `501`.
`gmlc_number`	string	`nil`	This GMLC's ISDN number, used as the MAP originator.
`hlr_gt`	string	`nil`	HLR global title for SendRoutingInfoForLC (serving MSC/SGSN discovery). Not neede if the request supplies `node_gt`.
`lcs_client_type`	atom	`:emergencyServices`	MAP LCS-ClientType fo the request.
`:omniss7 headless`	boolean	`true` (set in `config.exs`)	Run omniss7 as a library: codec + MAP transaction registry only.
`:omniss7 map_client_m3ua`	map	--	SCTP/M3UA association to the STP/SGP. Required for live operation.

> Accuracy on 2G/3G is Cell-ID / Timing-Advance class. The serving MSC/SGSN returns the serving cell, not a precise fix.

When `:omnilcs, :map enabled` is `true` and `:map_client_m3ua` is set, OmniLCS starts the M3UA association to the STP itself (supervised by

`OmniLcs.Control.MapStack`) — there is no separate transport step. Each MAP transaction is counted by the `omnilcs_map_request_total` Prometheus metric (tags `type`, `result`). If `enabled` is `true` but `:map_client_m3ua` is missing, the transport is skipped with a warning and the rest of OmniLCS is unaffected.

CAP Alerts

Configuration for CAP (Common Alerting Protocol) alert ingestion and broadcast. See the CAP Alert Operations Guide for full details on the alert lifecycle, polygon resolution, and operator approval workflow.

```
config :omnilcs, :cap,
  require_approval: true,
  plmn: %{mcc: "001", mnc: "01"},
  coverage_aware: false,
  feeds: []
```

Parameter	Type	Default	Description
`require_approval`	boolean	`true`	When `true`, alerts queue for operator approval. When `false`, alerts auto-broadcast.
`plmn`	map	`%{mcc: "001", mnc: "01"}`	PLMN identity (MCC/MNC) for broadcast messages
`coverage_aware`	boolean	`false`	Use cell coverage radius for polygon intersection (vs center-point only)
`feeds`	list	`[]`	CAP Atom feed URLs to poll

Feed Configuration

```
feeds: [
  %{url: "https://alerts.weather.gov/cap/us.php?x=1",
poll_interval_seconds: 60}
]
```

Field	Type	Required	Default	Description
`url`	string	Yes	--	CAP Atom feed URL
`poll_interval_seconds`	integer	No	`60`	Seconds between polls

Logger

```
config :logger,
  backends: [:console, ControlPanel.Logger]
```

Parameter	Type	Description
`backends`	list	Logger backends. `:console` writes to stdout; `ControlPanel.Logger` feeds the Log page in the web UI.

Complete Configuration Example

```
import Config

config :control_panel,
  parent_application: :omnilcs,
  parent_application_version: "1.0.0",
  parent_application_readable_name: "OmniLCS",
  home_page: ControlPanelWeb.ApplicationLive,
  use_builtin_pages: [
    {ControlPanelWeb.ApplicationLive, "/application",
"Resources"},
    {ControlPanelWeb.ConfigurationLive, "/configuration",
"Configuration"},
    {ControlPanelWeb.LogLive, "/log", "Log"}
  ],
  use_additional_pages: [
    {OmniLcs.Web.DashboardLive, "/dashboard", "Dashboard"},
    {OmniLcs.Web.LocationLive, "/location", "Location"},
    {OmniLcs.Web.CellDatabaseLive, "/cells", "Cells"},
    {OmniLcs.Web.DiameterLive, "/diameter", "Diameter"},
    {OmniLcs.Web.CbcLive, "/cbc", "CBC 2G"},
    {OmniLcs.Web.Cbc3gLive, "/cbc3g", "CBC 3G"},
    {OmniLcs.Web.Cbc4gLive, "/cbc4g", "CBC 4G"},
    {OmniLcs.Web.CapAlertsLive, "/cap", "CAP Alerts"}
  ],
  page_order: [
    "/dashboard", "/location", "/cells", "/diameter",
    "/cbc", "/cbc3g", "/cbc4g", "/cap", "/application",
"/configuration", "/log"
  ],
  licensee_name: "Omnitouch"

# REST API
config :api_ex,
  api: %{
    port: 8443,
    listen_ip: "0.0.0.0",
    product_name: "OmniLCS",
    title: "API - OmniLCS",
    hostname: "localhost",
    enable_tls: true,
```

```
    tls_cert_path: "priv/cert/omnitouch.crt",
    tls_key_path: "priv/cert/omnitouch.pem",
    routes: [
      %{path: "/status", module: OmniLcs.Api.StatusController,
actions: [:index]},
      %{path: "/location", module: OmniLcs.Api.LocationController,
actions: [:index, :create, :show]},
      %{path: "/cells", module: OmniLcs.Api.CellController,
actions: [:index, :create, :show, :update, :delete]},
      %{path: "/cap", module: OmniLcs.Api.CapController, actions:
[:index, :create, :show, :update]}
    ]
  }

# Control Panel HTTPS endpoint
config :control_panel, ControlPanelWeb.Endpoint,
  server: true,
  url: [host: "0.0.0.0", path: "/"],
  https: [port: 443, keyfile: "priv/cert/omnitouch.pem", certfile:
"priv/cert/omnitouch.crt"],
  adapter: Bandit.PhoenixAdapter,
  secret_key_base: "REPLACE_WITH_64_BYTE_RANDOM_SECRET",
  check_origin: false,
  pubsub_server: ControlPanel.PubSub,
  live_view: [signing_salt: "LcsLvSlt"]

# Logger
config :logger,
  backends: [:console, ControlPanel.Logger]

# E-SMLC general settings
config :omnilcs,
  esmlc_name: "OmniLCS",
  cell_database_path: "priv/cells.json"

# CBSP (2G Cell Broadcast) - BSCs connect to this port
config :omnilcs, :cbsp,
  listen_ip: "0.0.0.0",
  listen_port: 48049

# SABP (3G Cell Broadcast) - RNCs connect to this port
config :omnilcs, :sabp,
  listen_ip: "0.0.0.0",
  listen_port: 3452
```

```
# SLs (LCS-AP over SCTP) - E-SMLC connects to these MMEs for
positioning
config :omnilcs, :sls,
  local_ip: "10.5.198.200",
  mme_peers: [
    %{host: "mme01", ip: "10.179.1.15", port: 9082}
  ]

# SBC-AP (4G Cell Broadcast) - OmniLCS connects to these MMEs
config :omnilcs, :sbcap,
  local_ip: "10.5.198.200",
  mme_peers: [
    %{host: "mme01", ip: "10.179.2.100", port: 29168},
    %{host: "mme02", ip: "10.179.2.101", port: 29168}
  ]

# CAP Alert ingestion and broadcast
config :omnilcs, :cap,
  require_approval: true,
  plmn: %{mcc: "001", mnc: "01"},
  coverage_aware: false,
  feeds: []

# InfluxDB for cell position sync
config :omnilcs, OmniLcs.InfluxDb,
  database: "nokia-monitor",
  host: "172.19.3.68",
  port: 8086,
  auth: [method: :basic, username: "monitor", password:
"REPLACE_WITH_PASSWORD"],
  http_opts: [recv_timeout: 30_000],
  pool: [max_overflow: 10, size: 5]

# Diameter (SLg interface to DRA/MME - GMLC role, not E-SMLC SLs)
config :diameter_ex,
  diameter: %{
    service_name: :omnitouch_esmlc,
    listen_ip: "10.5.198.200",
    listen_port: 3868,
    host: "amanaki",
    realm: "epc.mnc380.mcc313.3gppnetwork.org",
    product_name: "OmniLCS",
    request_timeout: 5000,
```

```
    peer_selection_algorithm: :random,
    allow_undefined_peers_to_connect: true,
    log_unauthorized_peer_connection_attempts: true,
    control_module: OmniLcs.Control.Diameter,
    processor_module: DiameterEx.Processor,
    auth_application_ids: [],
    acct_application_ids: [],
    vendor_id: 10415,
    supported_vendor_ids: [5535, 10415],
    applications: [
      %{
        application_name: :slg,
        application_dictionary: :diameter_gen_3gpp_slg,
        vendor_specific_application_ids: [
          %{vendor_id: 10415, auth_application_id: 16_777_264,
acct_application_id: nil}
        ]
      }
    ],
    peers: [
      %{
        host: "dra01.epc.mnc380.mcc313.3gppnetwork.org",
        realm: "epc.mnc380.mcc313.3gppnetwork.org",
        ip: "10.179.2.233",
        port: 3868,
        tls: false,
        transport: :diameter_sctp,
        initiate_connection: true
      }
    ]
  }
```

GMLC & Le Interface Operations Guide

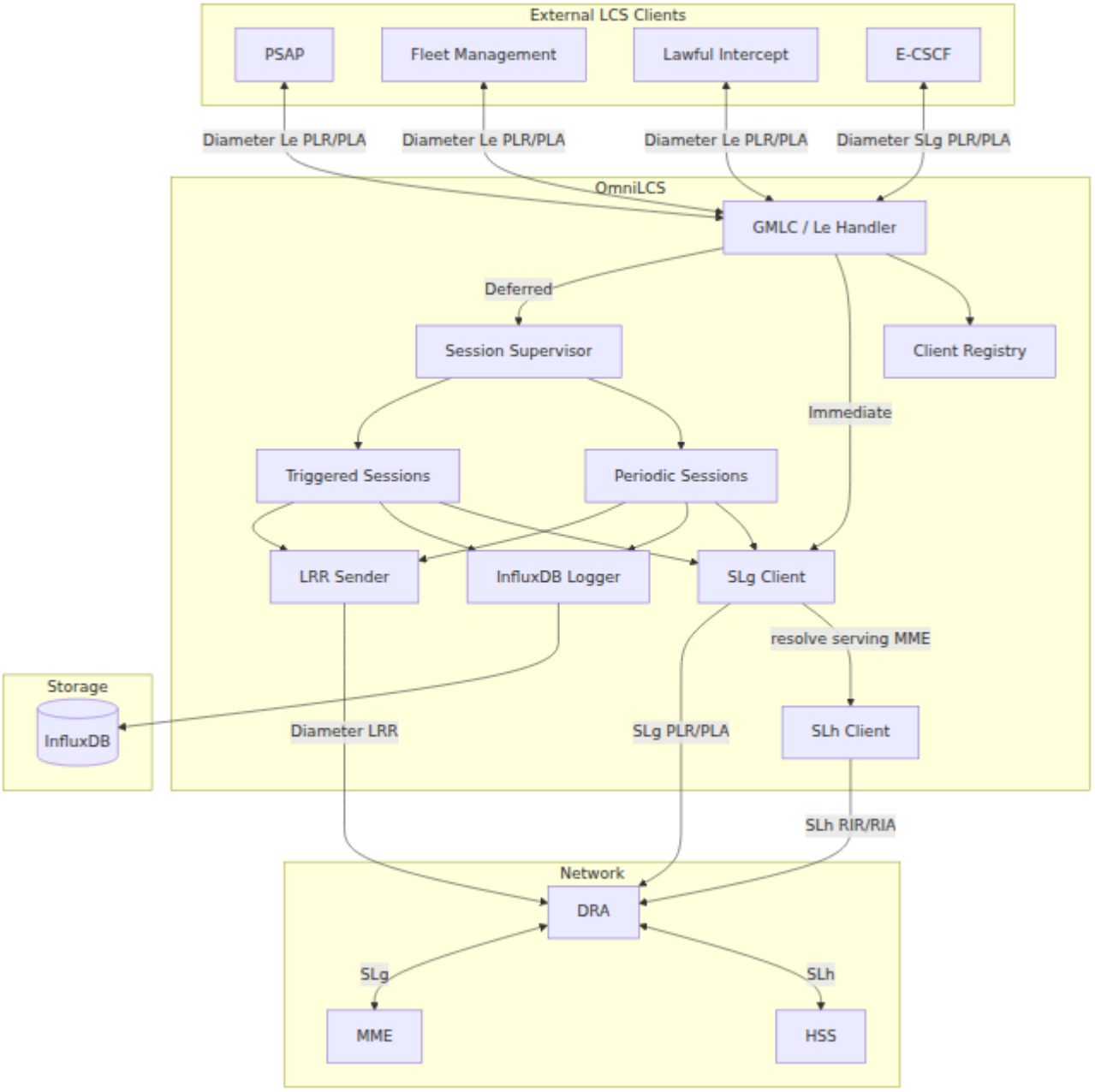
OmniLCS includes a Gateway Mobile Location Centre (GMLC) function that exposes the **Le interface** to external LCS clients. This allows Public Safety Answering Points (PSAPs), fleet management systems, lawful intercept platforms, and other external parties to request subscriber locations via the standard Diameter-based Le reference point defined in 3GPP TS 29.172.

The GMLC supports both **immediate** and **deferred** location requests. Deferred requests include **periodic location** (position reported at regular intervals) and **triggered location** (position reported when a subscriber enters, leaves, or remains inside a geographic area).

To fulfil a request the GMLC follows the standard 3GPP procedure: it discovers the subscriber's serving MME over the **SLh** interface (`LCS-Routing-Info-Request` to the HSS, per TS 29.173), then sends a **Provide-Location-Request (PLR)** to that MME over the **SLg** interface. The MME performs the positioning (directly, or by delegating to an E-SMLC over SLs/LCS-AP) and returns the `Location-Estimate` in the Provide-Location-Answer (PLA). In this **GMLC role** OmniLCS does not position the UE itself — it queries the serving MME and relays the result.

> This is one of several location options OmniLCS offers. For active, higher-accuracy positioning it uses its **E-SMLC role** over SLs instead, driving LPP/LPPa to the UE and eNB — see Two LCS roles, one location request. For subscribers on **2G/3G** the GMLC reaches the serving MSC/SGSN over **MAP/Lg** instead of Diameter SLg — see 2G/3G location over MAP/Lg. The same `POST /api/location` call reaches every path; the GMLC/SLg path described here is selected for 4G `cell`/last-known requests.

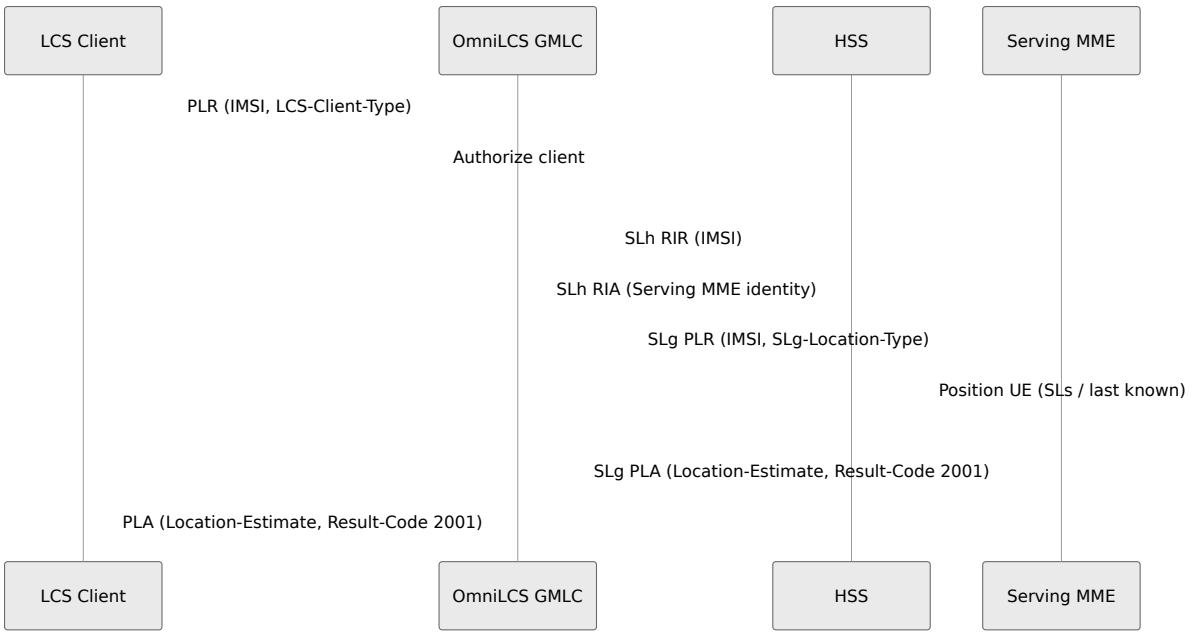
Architecture



Request Flow

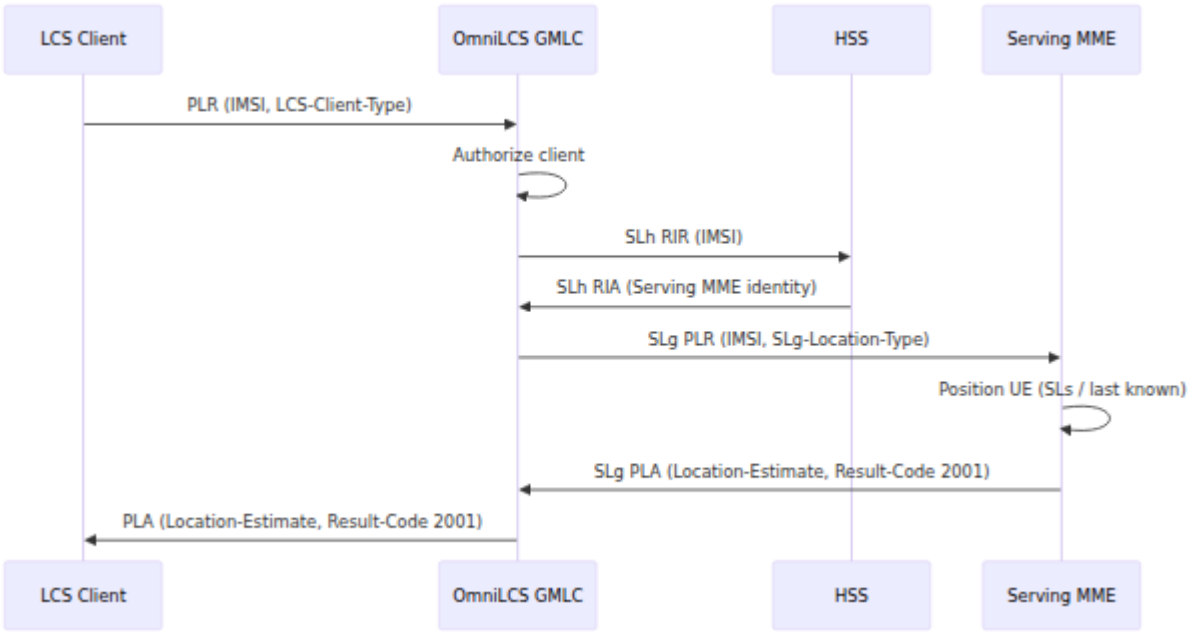
Immediate Location Request

An external LCS client sends a Provide-Location-Request (PLR) via Diameter. OmniLCS authorizes the client, discovers the serving MME over SLh, sends a PLR to that MME over SLg, and returns the MME's result in a Provide-Location-Answer (PLA).



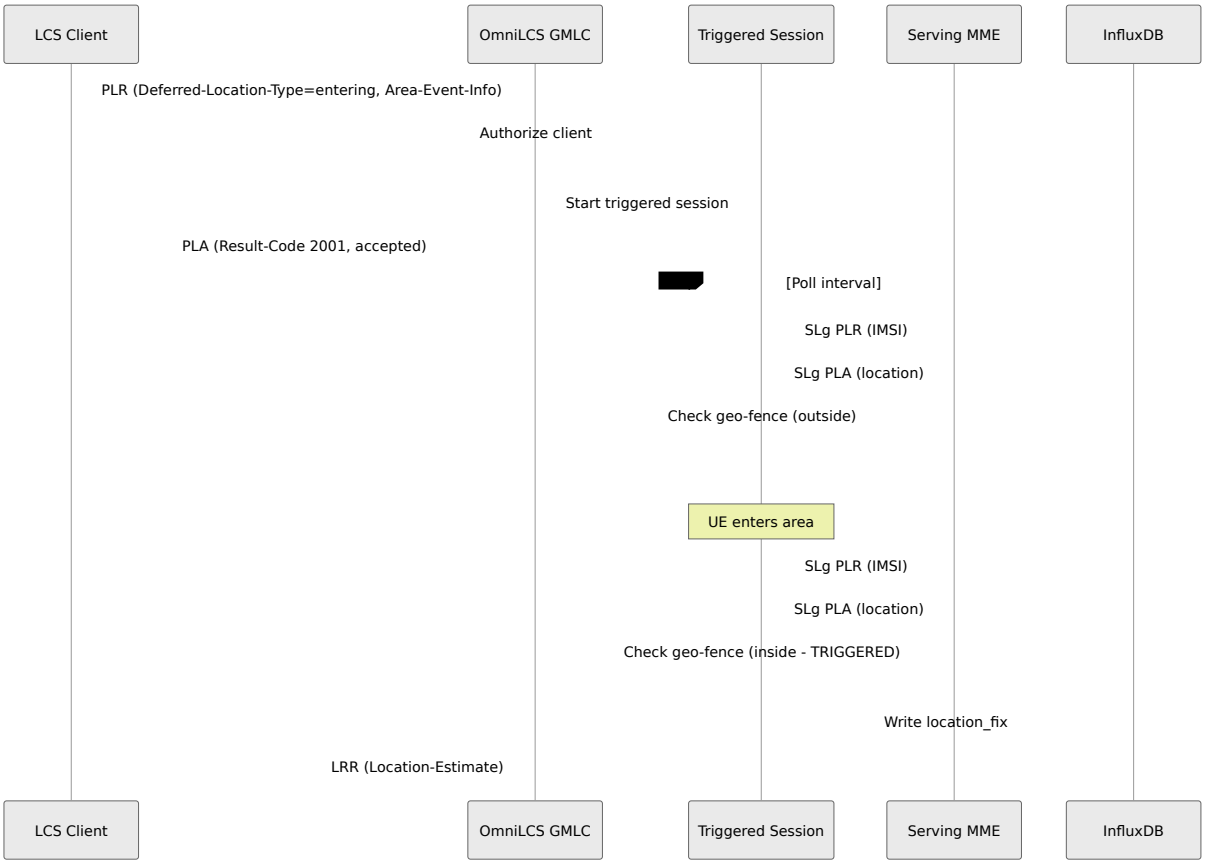
Periodic Location Request

When the PLR contains a `Deferred-Location-Type` with the periodic bit set and a `Periodic-LDR-Information` grouped AVP, OmniLCS creates a periodic session that runs location fixes at the specified interval and delivers each result via a Location-Report-Request (LRR).



Triggered (Geo-fence) Location Request

When the PLR contains area event information, OmniLCS creates a triggered session that polls the UE position and fires when the trigger condition is met.

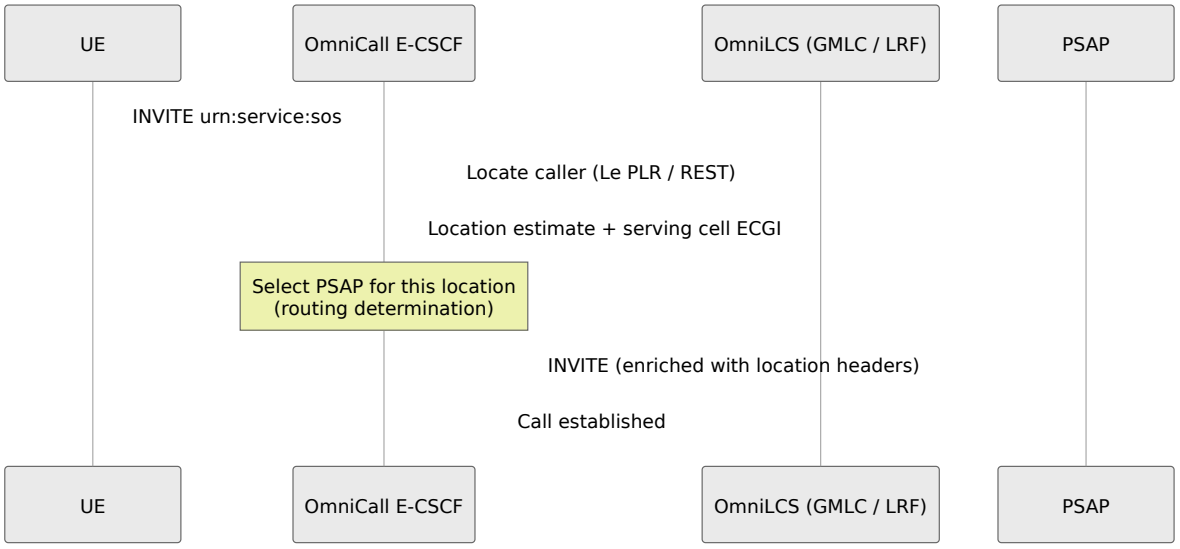


Emergency Call Routing with OmniCall (IMS)

The primary consumer of the GMLC is **OmniCall**, the IMS core. When a subscriber dials an emergency number, OmniCall's **E-CSCF** (Emergency Call Session Control Function) must route the call to the **PSAP** that serves the caller's physical location — routing to the wrong PSAP costs critical time. OmniCall gets that location from OmniLCS, which fills the **LRF / RDF** (Location Retrieval Function / Routing Determination Function) role of 3GPP TS 23.167.

The flow:

1. The UE places an IMS emergency call (`urn:service:sos`); it reaches OmniCall's E-CSCF.
2. OmniCall asks OmniLCS for the caller's location — over the **Le interface** (Diameter PLR, using the `:emergency_services` client type) or the REST API. OmniLCS answers through whichever role fits — GMLC/SLg for the serving cell, or E-SMLC/SLs for an active GNSS/OTDOA fix (see Two LCS roles, one location request).
3. OmniLCS returns the position (coordinates + serving cell ECGI).
4. OmniCall maps that location to the serving PSAP (routing determination) and forwards the INVITE there.
5. OmniCall **enriches the SIP signalling** so the PSAP receives the location with the call.



SIP location header enrichment

Using the position returned by OmniLCS, OmniCall adds the standard IMS-emergency location elements to the outbound INVITE, so the PSAP (and any intermediary) receives the caller's location with the call rather than having to query for it separately:

SIP element	Standard	Carries
`Geolocation` header + PIDF-LO body	RFC 6442 / RFC 4119	Geodetic (lat/lon + uncertainty) and/or civic location, by value or by reference
`P-Access-Network-Info`	RFC 7315	Serving access network and cell identity (ECGI)
`Geolocation-Routing`	RFC 6442	Whether downstream proxies may route on the conveyed location

The PSAP call-taker therefore sees the caller's location on call arrival, derived from the same OmniLCS positioning used everywhere else — the serving cell for a fast fix, or an active GNSS/OTDOA position when the UE and radio support it.

Configuration

GMLC Configuration

```
config :omnilcs, :gmlc,
  # Enable/disable the GMLC Le interface
  enabled: true,

  # Allow requests from clients not in the authorized list
  allow_unknown_clients: false,

  # Authorized external LCS clients
  authorized_clients: [
    %{
      name: "psap-01",
      type: :emergency_services,
      allowed_methods: [:cell, :ecid, :gnss, :otdoa],
      rate_limit: 100,
      description: "Primary PSAP"
    },
    %{
      name: "fleet-mgmt",
      type: :value_added_services,
      allowed_methods: [:cell, :ecid],
      rate_limit: 50,
      description: "Fleet management system"
    }
  ],

  # Allow deferred (periodic/triggered) location requests
  allow_deferred: true,

  # Maximum concurrent periodic sessions
  max_periodic_sessions: 100,

  # Maximum concurrent triggered (geo-fence) sessions
  max_triggered_sessions: 50,

  # Default interval for periodic sessions (ms)
  default_periodic_poll_interval_ms: 60_000,

  # Default polling interval for triggered sessions (ms)
```

```
  default_triggered_poll_interval_ms: 30_000,

  # Log all location fixes to InfluxDB
  influx_logging: true
```

GMLC Parameters

Parameter	Type	Required	Default	
`enabled`	Boolean	No	`false`	
`allow_unknown_clients`	Boolean	No	`false`	
`authorized_clients`	List	No	`[]`	
`allow_deferred`	Boolean	No	`true`	
`max_periodic_sessions`	Integer	No	`100`	

Parameter	Type	Required	Default	
`max_triggered_sessions`	Integer	No	`50`	
`default_periodic_poll_interval_ms`	Integer	No	`60000`	
`default_triggered_poll_interval_ms`	Integer	No	`30000`	
`influx_logging`	Boolean	No	`true`	

Client Parameters

Each entry in `authorized_clients`:

Parameter	Type	Required	Default	Description
`name`	String	Yes	--	Client identity. Matche against the LCS-EPS-Cl Name or LCS-Requesto String AVP in the PLR.
`type`	Atom	No	`:any`	Expected LCS-Client-Ty `:emergency_services` `:value_added_service` `:plmn_operator_servi` `:lawful_intercept_se` or `:any`.
`allowed_methods`	List	No	`[:cell, :ecid, :gnss, :otdoa]`	Positioning methods th may request.
`rate_limit`	Integer	No	`0`	Maximum requests per minute. `0` means unlin
`description`	String	No	`""`	Human-readable descr for the control panel.

LCS Client Types

Per 3GPP TS 29.172 section 7.4.4:

Value	Type	Description
0	Emergency Services	PSAP, E-CSCF emergency call routing
1	Value Added Services	Fleet management, asset tracking, location-based services
2	PLMN Operator Services	Operator internal services (O&M, network optimization)
3	Lawful Intercept Services	Law enforcement location requests

Deferred Location Types

Periodic Location

A periodic location session performs a configurable number of position fixes at a regular interval. Each fix is:

1. Performed by sending an SLg PLR to the subscriber's serving MME (resolved over SLh)
2. Written to InfluxDB as a `location_fix` measurement with session metadata
3. Logged via the standard location logger (CSV + ETS)
4. Delivered to the originating LCS client via Diameter LRR

The session terminates when the reporting count is exhausted.

Diameter AVPs used:

AVP	Code	Description
`Deferred-Location-Type`	1480	Bit 4 set indicates periodic LDR
`Periodic-LDR-Information`	2025	Grouped AVP containing reporting parameters
`Reporting-Amount`	2026	Number of position fixes to perform
`Reporting-Interval`	2027	Interval between fixes in seconds

Triggered (Geo-fence) Location

A triggered location session polls the UE position and evaluates it against one or more geographic areas. When the trigger condition is met, the position is reported.

Event types:

Event	Deferred-Location-Type Bit	Trigger Condition
Entering	Bit 1	UE transitions from outside to inside the area
Leaving	Bit 2	UE transitions from inside to outside the area
Being Inside	Bit 3	UE is inside the area on each poll

Supported area definitions:

- **Circle** -- Center point (latitude, longitude) and radius in meters
- **Polygon** -- List of vertices defining a closed polygon

The geo-fence evaluator reuses the ray-casting algorithm from the CAP alert polygon resolver for polygon containment checks, and haversine distance for circular area checks.

InfluxDB Location Logging

All GMLC session fixes are written to InfluxDB using the `location_fix` measurement:

Tags:

Tag	Description
`imsi`	Subscriber IMSI
`method`	Positioning method used (cell, ecid, gnss, otdoa)
`source`	Position source from the engine result
`session_type`	`periodic`, `triggered`, or `immediate`
`client_name`	Name of the requesting LCS client

Fields:

Field	Type	Description
`latitude`	Float	WGS84 latitude in degrees
`longitude`	Float	WGS84 longitude in degrees
`altitude`	Float	Altitude in meters (when available)
`uncertainty`	Float	Position uncertainty in meters
`confidence`	Integer	Confidence level (0-100)
`duration_ms`	Integer	Time taken to perform the fix

Example InfluxQL queries:

```
-- Latest fixes for a subscriber
SELECT * FROM location_fix WHERE imsi = '001010000000001' ORDER BY
time DESC LIMIT 10

-- Periodic session track over the last hour
SELECT latitude, longitude FROM location_fix
WHERE session_type = 'periodic' AND imsi = '001010000000001' AND
time > now() - 1h

-- Fix success rate by method
SELECT COUNT(*) FROM location_fix WHERE time > now() - 24h GROUP
BY method

-- Geo-fence trigger events
SELECT * FROM location_fix WHERE session_type = 'triggered' AND
time > now() - 24h
```

REST API

The deferred location API is available at `https://<host>:8443/api/deferred_location`.

List Active Sessions

```
GET /api/deferred_location
```

Response:

```
{
  "status": "ok",
  "count": 2,
  "data": [
    {
      "session_id": "a1b2c3d4-...",
      "type": "periodic",
      "imsi": "001010000000001",
      "method": "cell",
      "client_name": "rest-api",
      "status": "active",
      "interval_ms": 60000,
      "remaining_reports": 7,
      "total_reports": 10,
      "started_at": "2026-04-09T10:00:00Z",
      "last_fix_at": "2026-04-09T10:03:00Z"
    }
  ]
}
```

Create Periodic Session

```
POST /api/deferred_location
Content-Type: application/json

{
  "type": "periodic",
  "imsi": "001010000000001",
  "method": "cell",
  "interval_seconds": 60,
  "count": 10
}
```

Create Triggered Session

```
POST /api/deferred_location
Content-Type: application/json

{
  "type": "triggered",
  "imsi": "001010000000001",
  "method": "cell",
  "event_type": "entering",
  "poll_interval_seconds": 30,
  "max_reports": 0,
  "areas": [
    {
      "type": "circle",
      "center": {"lat": -33.8688, "lon": 151.2093},
      "radius_meters": 500
    }
  ]
}
```

Cancel Session

```
DELETE /api/deferred_location/:session_id
```

Metrics

Le Interface Requests

Metric: `omnilcs_gmlc_le_request_total` **Type**: Counter **Description**: Total Le interface requests from external LCS clients **Labels**:

- `client_type` -- LCS client type (emergency_services, value_added_services, etc.)
- `result` -- Request outcome: `received`, `success`, `error`, `unauthorized`

Metric: `omnilcs_gmlc_le_request_duration` **Type**: Histogram **Description**: Le request processing time in milliseconds **Labels**:

- `client_type` -- LCS client type

MAP / Lg Requests (2G/3G)

Metric: `omnilcs_map_request_total` **Type**: Counter **Description**: Total MAP/Lg location requests for 2G/3G subscribers (SendRoutingInfoForLCS, ProvideSubscriberLocation) **Labels**:

- `type` -- MAP operation (`MAP-PSL`)
- `result` -- `sent`, `success`

Location Report Requests

Metric: `omnilcs_gmlc_lrr_total` **Type**: Counter **Description**: Total Location Report Requests sent to external clients **Labels**:

- `session_type` -- `periodic` or `triggered`
- `result` -- `sent` or `error`

Session Tracking

Metric: `omnilcs_gmlc_session_periodic_active` **Type**: Gauge **Description**: Number of active periodic location sessions

Metric: `omnilcs_gmlc_session_triggered_active` **Type**: Gauge **Description**: Number of active triggered/geo-fence location sessions

Metric: `omnilcs_gmlc_session_total` **Type**: Counter **Description**: Total deferred sessions created **Labels**:

- `type` -- `periodic` or `triggered`

Geo-fence Events

Metric: `omnilcs_gmlc_geofence_trigger_total` **Type**: Counter **Description**: Total geo-fence trigger events fired **Labels**:

- `event_type` -- `entering`, `leaving`, or `being_inside`

InfluxDB Writes

Metric: `omnilcs_gmlc_influx_write_total` **Type**: Counter **Description**: Total InfluxDB location fix writes from GMLC sessions **Labels**:

- `result` -- `success` or `error`

Example Prometheus queries:

```
# Le request rate
rate(omnilcs_gmlc_le_request_total[5m])

# Deferred session creation rate by type
rate(omnilcs_gmlc_session_total[5m])

# Geo-fence trigger rate
rate(omnilcs_gmlc_geofence_trigger_total[5m])

# LRR delivery error ratio
sum(rate(omnilcs_gmlc_lrr_total{result="error"}[5m]))
  / sum(rate(omnilcs_gmlc_lrr_total[5m]))
```

Troubleshooting

Le Requests Returning 5012 (UNABLE_TO_COMPLY)

Symptoms: External LCS client receives result code 5012 for all PLRs.

Possible causes:

- GMLC not enabled in config (`enabled: false`)
- Client not in `authorized_clients` and `allow_unknown_clients` is `false`
- SLh resolution failed or there is no SLg route to the serving MME (check DRA/HSS/MME connectivity)

Resolution:

1. Verify `config :omnilcs, :gmlc, enabled: true` is set
2. Check the client name matches the `LCS-EPS-Client-Name` AVP in the PLR
3. Verify the SLh/SLg Diameter peers (DRA, HSS, MME) are connected on the Dashboard

Periodic Sessions Not Starting

Symptoms: PLR with periodic deferred type returns 3004 (TOO_BUSY).

Possible causes:

- `max_periodic_sessions` limit reached
- Session supervisor not running

Resolution:

1. Check active session count on the GMLC control panel page
2. Increase `max_periodic_sessions` if needed
3. Verify `OmniLcs.Gmlc.SessionSupervisor` is running in the Application page

Location Fixes Not Appearing in InfluxDB

Symptoms: Periodic/triggered sessions are running but no data in InfluxDB.

Possible causes:

- `influx_logging` set to `false`
- InfluxDB connection down
- Database does not exist

Resolution:

1. Verify `influx_logging: true` in GMLC config
2. Check InfluxDB connectivity (cell sync status is a good proxy)
3. Query InfluxDB directly: `SHOW MEASUREMENTS` should include `location_fix`

Geo-fence Not Triggering

Symptoms: Triggered session is active but never fires.

Possible causes:

- Area definition too small for the positioning method's accuracy
- Wrong event type (e.g., `:leaving` when subscriber is already outside)
- Positioning method returning `nil` coordinates

Resolution:

1. Check the positioning method accuracy -- Cell ID has kilometre-scale uncertainty, which may be too coarse for small geo-fences
2. Use a more accurate method (`:ecid` or `:gnss`) for smaller areas
3. Verify the area coordinates are correct (latitude/longitude order)

E-SMLC Location Services Guide

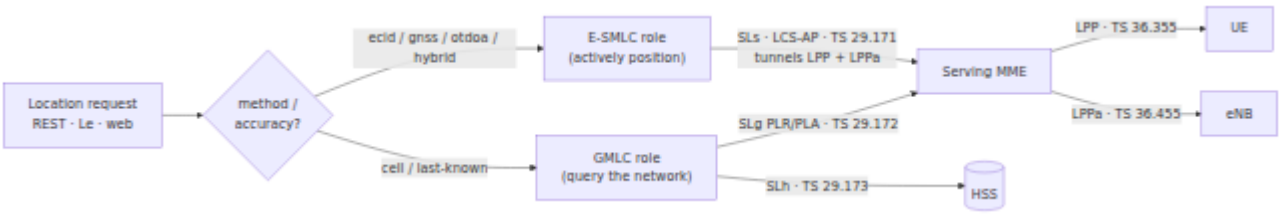
OmniLCS implements an Evolved Serving Mobile Location Centre (E-SMLC) that provides UE positioning for LTE networks. The E-SMLC determines UE location using multiple positioning methods and communicates with the MME via the SLs interface using native LCS-AP over SCTP per 3GPP TS 29.171.

Two LCS roles, one location request

OmniLCS is co-located as both an **E-SMLC** and a **GMLC** — the two location network functions defined by the 3GPP LTE LCS architecture (TS 23.271). A location client never chooses between them: it asks for a UE's location over the REST API, the Location web page, or the Le interface, and OmniLCS picks the role and interface from the requested `method`/`accuracy`. Both options are always available behind the one "get location" call.

- **Active / high-accuracy methods** (E-CID, GNSS, OTDOA, hybrid) use the **E-SMLC** role documented in this guide. OmniLCS drives **LPP** (to the UE, TS 36.355) and **LPPa** (to the eNB, TS 36.455) over the **SLs / LCS-AP** interface (TS 29.171), relayed by the serving MME, for a precise fix.
- **Cell ID / last-known** requests use the **GMLC** role (see the GMLC & Le Interface guide). OmniLCS resolves the serving MME over **SLh** (TS 29.173) and sends a Provide-Location-Request over **SLg** (TS 29.172); the MME returns the serving cell from the UE's context.

In both roles the **serving MME is the anchor**: it is the only node with a signalling path to the UE (NAS) and the eNB (S1AP), so it is either the SLg responder or the SLs relay for LPP/LPPa. The GMLC, with only SLh/SLg Diameter links, cannot reach the radio itself — which is exactly why active positioning is the E-SMLC's job.



See the REST API reference for the harmonised request/response schema and the full decision table. The REST API selects the path per request; the Le Diameter interface and deferred (periodic/triggered) sessions currently always use the GMLC/SLg path.

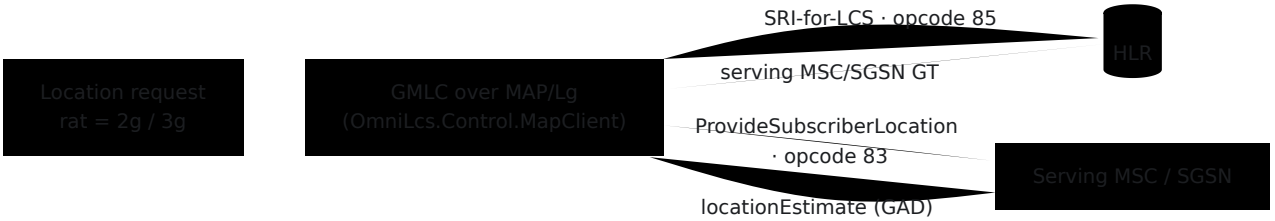
2G/3G location over MAP/Lg

For subscribers on **GERAN (2G)** or **UTRAN (3G)** there is no Diameter SLg/SLs — the GMLC reaches the legacy core over the **Lg** interface using **MAP** (TS 29.002) on an SS7/SIGTRAN stack (SCTP → M3UA → SCCP → TCAP). OmniLCS provides this through the `OmniLcs.Control.MapClient` module, backed by the `omniss7` SS7 stack, as a third option behind the same `/api/location` call (selected with `rat: "2g"` / `"3g"`).

The flow is a two-step MAP transaction:

1. **SendRoutingInfoForLCS** (opcode 85) to the **HLR** → returns the subscriber's serving **MSC/SGSN** global title. (Skipped if a `node_gt` is supplied.)
2. **ProvideSubscriberLocation** (opcode 83) to that **MSC/SGSN** → returns the location estimate.

The `locationEstimate` is the **same TS 23.032 GAD** shape as 4G, so it is decoded with the shared `decode_gad/1` and returned in the standard location map with `source: "map"`.

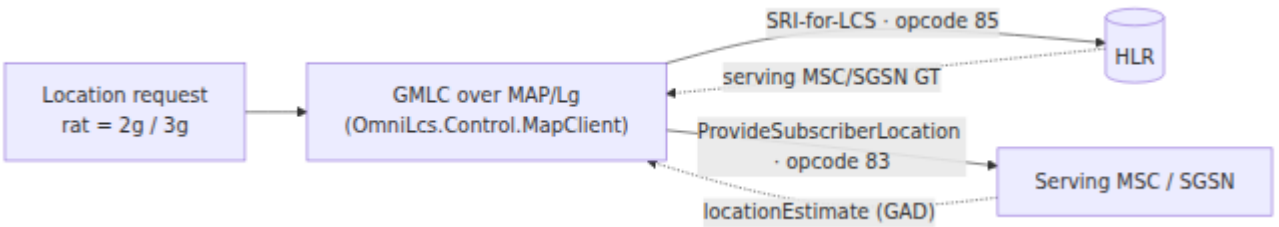


Accuracy on 2G/3G is **Cell-ID / Timing-Advance class** — the network returns the serving cell, not a precise fix. This path is disabled by default (`config :omnilcs, :map, enabled: false`); enabling it requires the GMLC number, HLR

global title, and a configured SS7 peer (STP/MSC/SGSN) over `omniss7`'s `:map_client_m3ua` transport. See the GMLC guide and Configuration.

Operation. `omniss7` runs **headless** (library mode — just the MAP codec and transaction registry), so when `:omnilcs, :map` is enabled OmniLCS starts the **M3UA association to the STP itself** (supervised by `OmniLcs.Control.MapStack`); when disabled it starts nothing. Each MAP transaction increments the `omnilcs_map_request_total` Prometheus counter (tags `type` = `MAP-PSL`, `result` = `sent`/`success`). A request with `rat: "2g"`/`"3g"` while the path is disabled returns HTTP `501`; an unresolved serving node returns `422`.

Architecture



Positioning Methods

OmniLCS supports the full range of LTE positioning techniques. A request names a `method` (or a target `accuracy`, which auto-selects one); OmniLCS runs that technique on the appropriate interface and returns the estimate. Every method maps to a standard 3GPP positioning procedure:

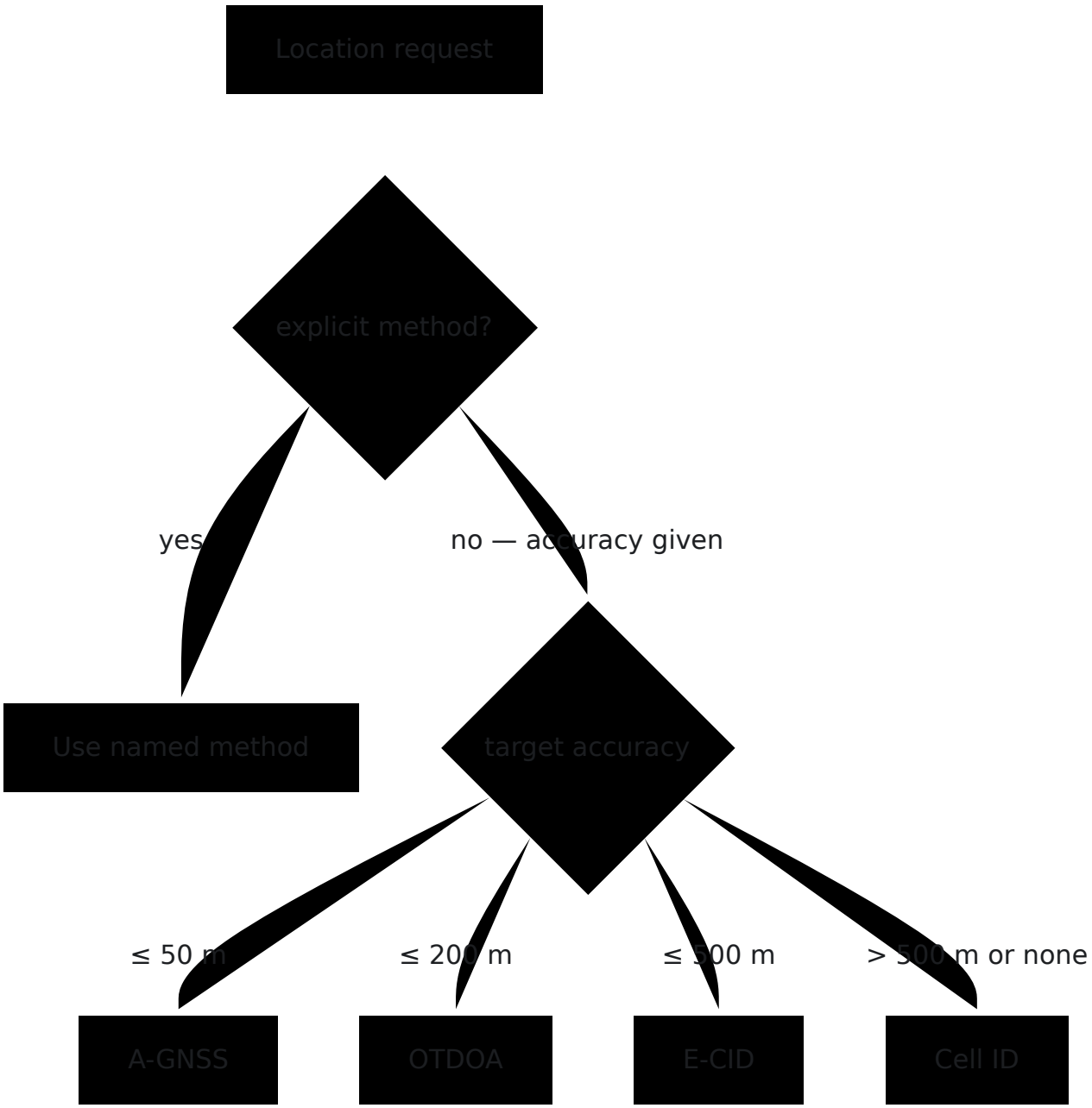
Method	Technique	Positioning protocol	Role / interface	Typical accuracy	Ne
Cell ID (CID)	Serving-cell position	cell DB lookup, or SLg PLR	E-SMLC (local) **or** GMLC (SLg)	cell radius, 100 m – 5 km	not
Enhanced Cell ID (E-CID)	Single-cell ranging: timing advance + RSRP/RSRQ	LPPa (eNB) → LPP (UE) fallback	E-SMLC / SLs	50 m – 500 m	eNI UE
OTDOA	Downlink multilateration ("triangulation") from PRS time-differences across cells	LPPa (PRS config) + LPP (RSTD)	E-SMLC / SLs	10 m – 100 m	UE eNI PRS
A-GNSS	Assisted satellite (GPS/Galileo/…) fix in the UE	LPP (UE)	E-SMLC / SLs	5 m – 50 m	UE GN
Hybrid	Combine A-GNSS + E-CID in one request, best-of	LPP (UE)	E-SMLC / SLs	best available	UE

> **What "triangulation" means here.** True triangulation/multilateration of an LTE UE is **OTDOA**: the UE measures the time difference of downlink Positioning Reference Signals (PRS) from several cells and OmniLCS solves the resulting hyperbolae for a position (see OTDOA Calculation). E-CID is the coarser single-cell cousin — range from one cell's timing advance rather than multi-cell geometry. Uplink multilateration (**U-TDOA**, network-measured) is **not** implemented: it requires Location Measurement Units (LMUs) not present in an EPC-only deployment.

Choosing a method by accuracy

If a request supplies `accuracy` (in metres) instead of an explicit `method`, OmniLCS picks the lightest method that can plausibly meet the target:

Requested accuracy	Method selected
≤ 50 m	A-GNSS
≤ 200 m	OTDOA
≤ 500 m	E-CID
> 500 m	Cell ID



Cell ID

The simplest positioning method. Returns the geographic position of the serving cell from the cell database.

- **Accuracy**: Cell radius (typically 100m - 5km depending on cell type)
- **Latency**: Milliseconds (local database lookup)
- **UE Support Required**: No
- **eNB Support Required**: No

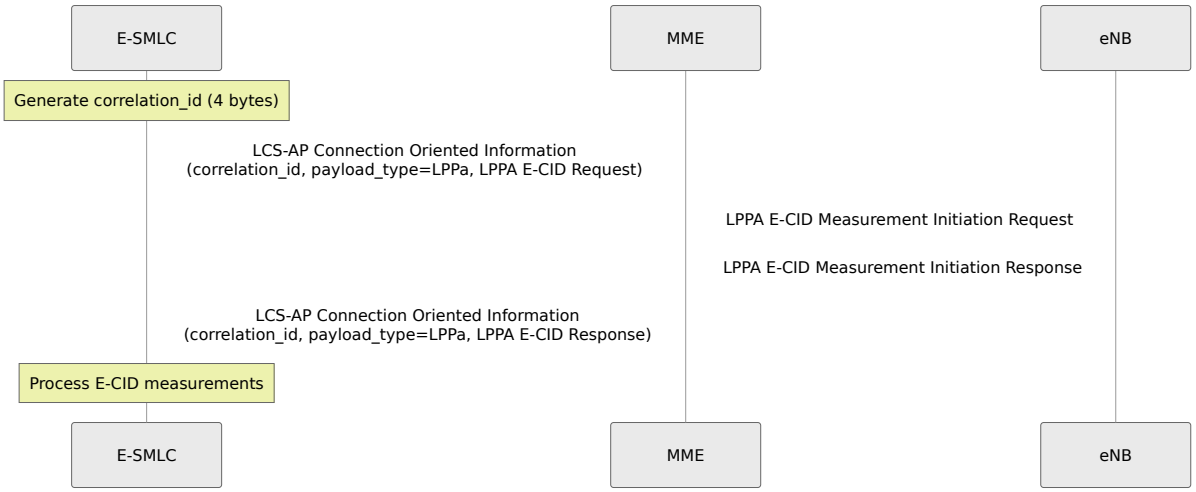
The engine first attempts to look up the serving cell in the local cell database. If not found, it sends an LPPA E-CID Measurement Initiation Request (with cell_id quantity only) to the eNB via the MME using LCS-AP Connection Oriented Information to obtain the cell identity.

The same serving cell can also be obtained **without touching the radio** via the GMLC role — an SLg Provide-Location-Request to the serving MME, which returns the ECGI from the UE's context. This is the path used for `cell`/last-known REST requests and all Le-interface requests (see Two LCS roles, one location request).

Enhanced Cell ID (E-CID)

Uses LPPa to obtain radio measurements from the eNB, including timing advance and signal strength. If the eNB rejects the LPPa E-CID procedure (e.g. `requested-item-not-supported`), OmniLCS falls back to requesting E-CID measurements directly from the **UE** over LPP.

- **Accuracy**: 50m - 500m
- **Latency**: 1-5 seconds
- **UE Support Required**: No for the LPPa path; Yes for the LPP fallback
- **eNB Support Required**: Yes for the LPPa path; the LPP fallback needs neither



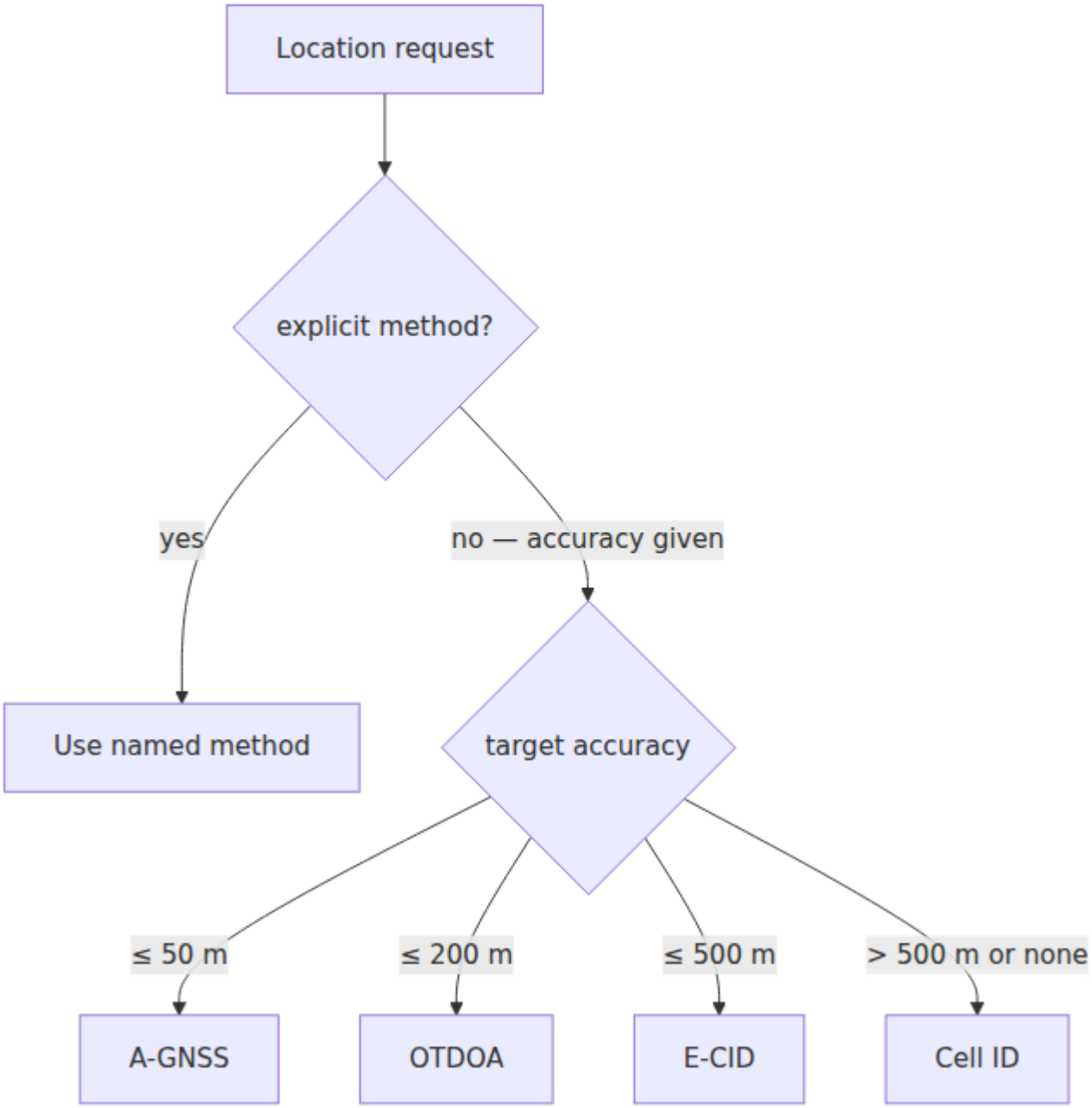
Measurement quantities requested:

Quantity	Description
Cell ID	Serving cell identity
Timing Advance Type 2	UE-to-eNB propagation delay (distance estimate)
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality

GNSS / A-GPS

Requests the UE to report its GPS coordinates via the LPP protocol.

- **Accuracy**: 5m - 50m
- **Latency**: 5-30 seconds
- **UE Support Required**: Yes (GNSS capability)
- **eNB Support Required**: No



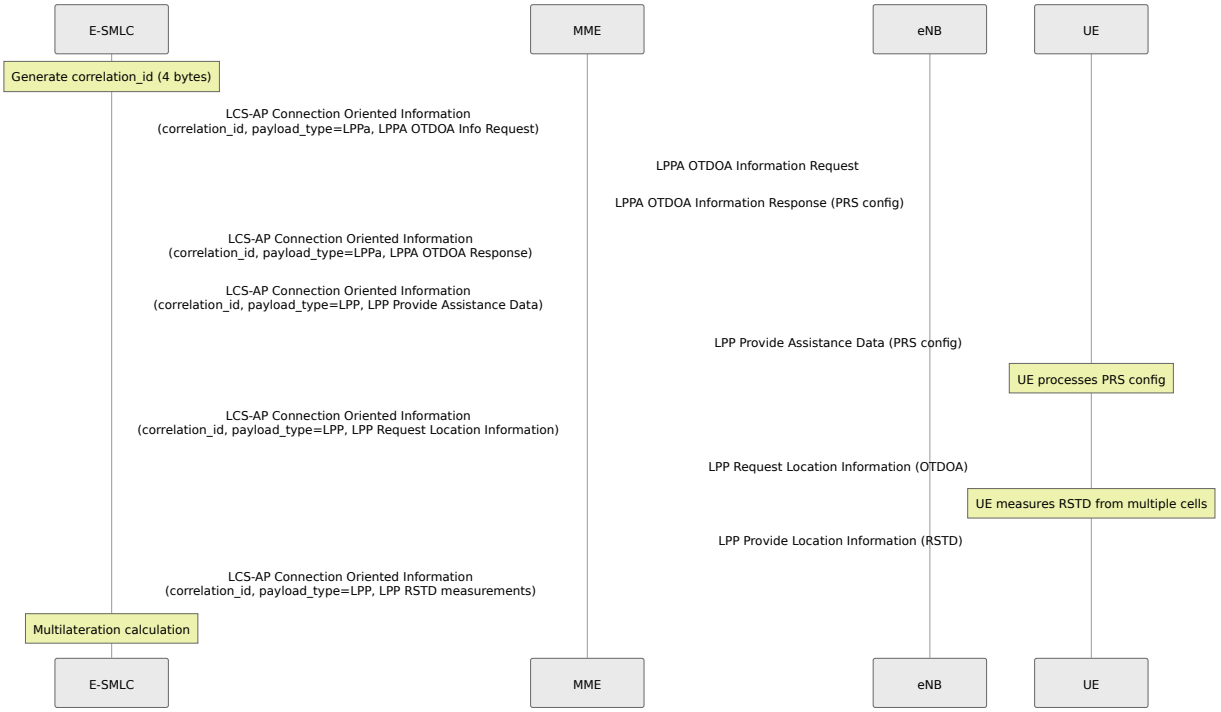
The E-SMLC sends an LPP Request Location Information message for GNSS to the UE through the MME via LCS-AP. The UE performs a GNSS fix and reports its coordinates back via LPP, relayed through the MME in a Connection Oriented Information message.

OTDOA

Observed Time Difference of Arrival. Uses multilateration from Positioning Reference Signal (PRS) measurements across multiple cells.

- **Accuracy**: 10m - 100m
- **Latency**: 5-15 seconds

- **UE Support Required**: Yes (OTDOA capability)
- **eNB Support Required**: Yes (PRS configured, LPPA)



OTDOA steps:

1. Request OTDOA cell information from eNB via LPPA (relayed through LCS-AP Connection Oriented Information)
2. Receive PRS configuration, cell IDs, EARFCNs for reference and neighbour cells
3. Send PRS assistance data to UE via LPP (relayed through LCS-AP Connection Oriented Information)
4. Request RSTD measurements from UE via LPP
5. Receive RSTD measurements
6. Run multilateration to compute position

Hybrid

Requests A-GNSS **and** E-CID in a single LPP transaction and returns the best available fix, falling back to the coarser E-CID measurement when the UE cannot complete a GNSS fix.

- **Accuracy**: Best of A-GNSS (5-50m) and E-CID (50-500m)
- **Latency**: 5-30 seconds (bounded by the GNSS fix)

- **UE Support Required**: Yes (GNSS + measurement reporting)
- **eNB Support Required**: No

Parse error on line 11: ...refer GNSS position;
fall back to E- ----------
------^ Expecting 'SPACE', 'NEWLINE', 'create', 'box', 'end', 'autonumber',
'activate', 'deactivate', 'title', 'legacy_title', 'acc_title', 'acc_descr',
'acc_descr_multiline_value', 'loop', 'rect', 'opt', 'alt', 'par', 'par_over', 'critical',
'break', 'option', 'and', 'else', 'participant', 'participant_actor', 'destroy', 'note',
'links', 'link', 'properties', 'details', 'ACTOR', got 'INVALID'

Try again

OmniLCS sends one combined LPP Request Location Information for GNSS + E-CID and merges whatever the UE returns, preferring the GNSS position. This minimises signalling versus running the two methods separately.

SLs Interface -- LCS-AP over SCTP

The E-SMLC communicates with the MME using the LCS Application Protocol (LCS-AP) over SCTP per 3GPP TS 29.171. OmniLCS initiates SCTP associations to each configured MME peer.

Transport Details

Parameter	Value
Protocol	LCS-AP over SCTP
SCTP PPID	29
Default port	9082 (IANA registered)
Connection direction	E-SMLC initiates to MME
Encoding	ASN.1 PER aligned

LCS-AP Procedures

Procedure	Code	Description
Location Request / Location Response	0	E-SMLC requests MME to locate a UE
Connection Oriented Information	1	Relay LPPA/LPP PDUs (UE-associated, uses correlation_id)
Connectionless Information	2	Relay non-UE-associated LPPA PDUs
Location Abort	3	Abort an in-progress location procedure
Reset	4	Reset all location procedures

Correlation ID

Every UE-associated LCS-AP transaction uses a **correlation ID** -- a 4-byte binary value that ties together the Location Request, all Connection Oriented

Information exchanges, and the Location Response for a single positioning session. The E-SMLC generates a random correlation ID at the start of each session using `:crypto.strong_rand_bytes(4)`.

The correlation ID is included as an IE (ID 2) in:

- Location Request (E-SMLC -> MME)
- Location Response (MME -> E-SMLC)
- Connection Oriented Information (both directions)
- Location Abort (E-SMLC -> MME)

APDU Tunneling

LPP and LPPA PDUs are carried inside LCS-AP **Connection Oriented Information** messages. The payload type IE (ID 15) identifies the content:

Payload Type	Value	Description
`:lPP`	0	LPP PDU (UE positioning protocol, TS 36.355)
`:lPPa`	1	LPPa PDU (eNB positioning protocol, TS 36.455)

The APDU IE (ID 1) carries the binary LPP or LPPa PDU. The MME transparently relays these between the E-SMLC and the target eNB or UE.

Inbound Messages (MME -> E-SMLC)

Location Response (procedure code 0, successfulOutcome / unsuccessfulOutcome)

The MME sends a Location Response as the outcome of a Location Request.

IE	ID	Criticality	Description
Correlation-ID	2	Reject	Correlation ID from the original Location Request
Location-Estimate	12	Reject	GAD-encoded position (on success)
Positioning-Data	16	Reject	Positioning method information (on success)
Velocity-Estimate	21	Reject	Velocity of the UE (optional, on success)
Accuracy-Fulfilment-Indicator	0	Reject	Whether requested accuracy was met (on success)
LCS-Cause	11	Ignore	Failure cause (on failure)

Connection Oriented Information (procedure code 1, initiatingMessage)

The MME relays LPPA/LPP PDUs from the eNB or UE back to the E-SMLC.

IE	ID	Criticality	Description
Correlation-ID	2	Reject	Session correlation
Payload-Type	15	Reject	`:lPP` or `:lPPa`
APDU	1	Reject	Binary LPP or LPPa PDU

Reset Request (procedure code 4, initiatingMessage)

The MME requests the E-SMLC to reset all active procedures. The E-SMLC responds with a Reset Acknowledge.

IE	ID	Criticality	Description
LCS-Cause	11	Ignore	Reason for reset

Outbound Messages (E-SMLC -> MME)

Location Request (procedure code 0, initiatingMessage)

The E-SMLC requests the MME to locate a UE.

IE	ID	Criticality	Description
Correlation-ID	2	Reject	4-byte session correlation ID
Location-Type	13	Reject	`geographic-Information`, `assistance-Information`, or `last-known-location`
E-UTRAN-Cell-Id	4	Ignore	E-CGI of the UE serving cell
LCS-Client-Type	8	Reject	Type of LCS client (optional)
LCS-Priority	9	Reject	Location request priority (optional)
LCS-QoS	10	Reject	Quality of service requirements (optional)
IMSI	7	Ignore	UE IMSI (optional)
IMEI	6	Ignore	UE IMEI (optional)
Include-Velocity	5	Reject	Request velocity estimate (optional)

Connection Oriented Information (procedure code 1, initiatingMessage)

The E-SMLC sends LPPA/LPP PDUs to the eNB or UE via the MME.

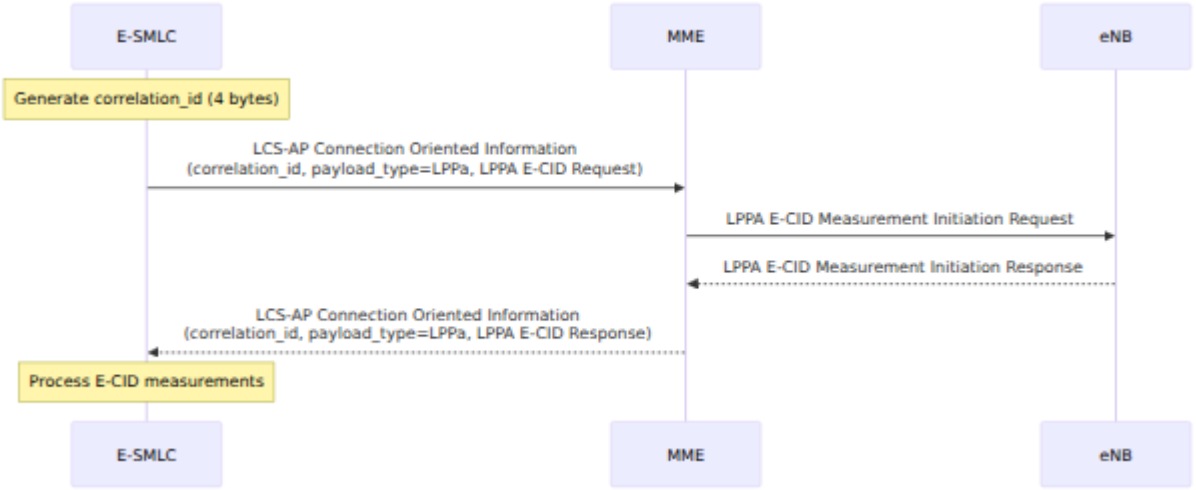
IE	ID	Criticality	Description
Correlation-ID	2	Reject	Session correlation
Payload-Type	15	Reject	`:lPP` or `:lPPa`
APDU	1	Reject	Binary LPP or LPPa PDU

Connectionless Information (procedure code 2, initiatingMessage)

The E-SMLC sends non-UE-associated LPPA PDUs (e.g., eNB position requests) through the MME.

IE	ID	Criticality	Description
Source-Identity	19	Reject	E-SMLC network element identity
Destination-Id	3	Reject	Target eNB network element identity
APDU	1	Reject	Binary LPPa PDU

Location Session Lifecycle



Each session tracks:

Field	Description
`session_id`	Unique identifier (e.g., `esmlc-1234567890-1`)
`imsi`	UE identifier
`mme_host`	MME that originated the request
`method`	Positioning method used
`state`	Current session state
`created_at`	Session creation timestamp
`updated_at`	Last state change timestamp
`completed_at`	Completion timestamp
`lppa_transactions`	List of LPPA PDUs exchanged
`lpp_transactions`	List of LPP PDUs exchanged
`measurements`	Accumulated measurement data
`result`	Final location result

Sessions older than 1 hour can be cleaned up via `LocationSession.cleanup_old_sessions/1`.

Pending Transaction Correlation

The E-SMLC uses an ETS table (`:pending_transactions`) to correlate outgoing LCS-AP messages with their responses. When a positioning session sends a Connection Oriented Information message:

1. The engine generates a 4-byte correlation ID

2. Registers `{correlation_id, {caller_pid, ref}}` in the `:pending_transactions` table
3. Sends the LCS-AP message via the SCTP transport
4. Waits in a `receive` block for the response
5. When the SLs handler receives a matching response, it looks up the pending transaction by correlation ID and sends the result to the waiting process

Cell Database Management

The cell database stores geographic positions and radio parameters for each cell site. It is used for Cell ID positioning and OTDOA calculations.

Cell Record Fields

Field	Type	Required	Description
`cell_id`	any	Yes	Unique cell identifier
`latitude`	float	Yes	Cell latitude in decimal degrees
`longitude`	float	Yes	Cell longitude in decimal degrees
`pci`	integer	No	Physical Cell Identity (0-503)
`earfcn`	integer	No	E-UTRA Absolute Radio Frequency Channel Number
`radius`	integer	No	Cell coverage radius in meters (default: 1000)
`azimuth`	float	No	Antenna azimuth in degrees
`height`	float	No	Antenna height in meters
`prs_config`	map	No	PRS configuration for OTDOA

PRS Configuration Fields

Field	Type	Description
`bandwidth`	integer	PRS bandwidth in resource blocks (6, 15, 25, 50, 75, 100)
`config_index`	integer	PRS configuration index (0-4095)
`num_dl_frames`	integer	Number of consecutive DL subframes
`cp_length`	atom	Cyclic prefix length (`:normal` or `:extended`)
`num_antenna_ports`	integer	Number of antenna ports (1, 2, or 4)

InfluxDB Cell Sync

Cell positions are periodically synchronized from InfluxDB:

Parameter	Value	Description
Sync interval	5 minutes	Automatic sync period
Initial delay	10 seconds	Delay before first sync after startup
Sync timeout	60 seconds	Maximum time for a sync operation

Sync can also be triggered manually via:

- REST API: `POST /api/cells/sync`
- LiveView UI: "Sync from InfluxDB" button on the Cells page

JSON Import

Cells can be imported from a JSON file:

```
[
  {
    "cell_id": "001-01-0001-01",
    "latitude": 40.7128,
    "longitude": -74.0060,
    "pci": 100,
    "earfcn": 1300,
    "radius": 500,
    "prs_config": {
      "bandwidth": 50,
      "config_index": 0,
      "num_dl_frames": 1
    }
  }
]
```


Nearby Cell Search

The cell database supports geographic proximity queries using the Haversine formula for great-circle distance calculation. Query via REST API: `GET /api/cells/nearby?lat=X&lon=Y&radius=R` where radius is in kilometers.

OTDOA Calculation

The OTDOA calculator converts RSTD (Reference Signal Time Difference) measurements into a UE position using multilateration.

Algorithm

1. **RSTD to distance difference**: `dd = RSTD * Ts * c` where Ts = 1/(15000 * 2048) seconds and c = speed of light
2. **Coordinate projection**: Cell lat/lon coordinates are projected to a local meter-based coordinate system
3. **Iterative least squares**: Solves the hyperbolic positioning equations using weighted least squares with Jacobian-based optimization
4. **Convergence**: Iterates until position change is less than 1 meter or maximum 50 iterations

5. **Uncertainty estimation**: Computed from measurement geometry and distance differences

Requirements

- Minimum 2 neighbour cells (plus reference cell = 3 total) for a position estimate
- 3 or more neighbours recommended for unambiguous 2D fix
- All cells must have known positions in the cell database
- Cell positions are resolved by PCI, ECGI, or cell_id

REST API Reference for Location

See REST API Reference for the full endpoint documentation. Key location endpoints:

Endpoint	Method	Description
`/api/location`	POST	Request a new UE location
`/api/location`	GET	List recent location fixes
`/api/location/:imsi`	GET	Last known location for an IMSI
`/api/location/:imsi/history`	GET	Location history for an IMSI
`/api/location/:imsi/history/csv`	GET	CSV export of location history

Troubleshooting

No SLs Peers Connected

1. Verify the `:sls` configuration: `local_ip` must be reachable from the MME network
2. Check that each `mme_peers` entry has the correct IP address and port (default 9082)
3. Look for SCTP connection errors in the log: `SLs: Failed to connect to MME`
4. Verify SCTP is not blocked by firewalls (IP protocol 132)
5. Confirm the MME is listening for LCS-AP on port 9082

Location Request Returns "no_mme_host"

The E-SMLC cannot determine which MME to send LPPA/LPP messages to. Ensure:

1. At least one SLs SCTP association is established
2. For REST API requests, provide the `mme_host` parameter
3. Check the SLs connection status on the Dashboard page

No LCS-AP Response from MME

1. Verify the SCTP association is in `:established` state via `SctpTransport.get_connections/0`
2. Check for SCTP heartbeat failures in the log
3. Confirm the MME supports LCS-AP (TS 29.171)
4. Check that the correlation ID is being correctly matched between request and response

Cell ID Positioning Returns No Coordinates

The cell database does not contain a matching cell. Actions:

1. Trigger an InfluxDB sync: `POST /api/cells/sync`

2. Add cells manually via the REST API or LiveView UI
3. Import cells from a JSON file

OTDOA Timeout

The eNB did not respond with OTDOA information within the timeout period. Possible causes:

1. eNB does not support LPPA OTDOA Information procedure
2. PRS is not configured on the eNB
3. Network path issue between MME and eNB

GNSS Timeout

The UE did not report GNSS coordinates within the timeout. Possible causes:

1. UE does not support GNSS positioning
2. UE is indoors (no satellite visibility)
3. No GNSS assistance data provided (cold start takes longer)

3GPP References

Specification	Title
TS 29.171	LCS Application Protocol (LCS-AP) between MME and E-SMLC (SLs interface)
TS 29.172	EPC LCS Protocol between GMLC and MME (SLg Diameter interface)
TS 36.455	LTE Positioning Protocol A (LPPa) between eNB and E-SMLC
TS 36.355	LTE Positioning Protocol (LPP) between UE and E-SMLC
TS 23.032	Universal Geographical Area Description (GAD encoding)

OmniLCS REST API Reference

The OmniLCS REST API is served over HTTPS on port **8443**. The base URL is `https://<host>:8443/api`.

API documentation (Swagger UI) is available at `https://<host>:8443/api/docs`. The OpenAPI schema is at `https://<host>:8443/api/schema`.

Authentication

The API does not currently enforce authentication. Access should be restricted at the network level.

Response Format

All responses use JSON. Successful responses include a `"status": "ok"` field. Error responses include `"status": "error"` and a `"reason"` field.

System Status

GET /api/status

Returns system health and operational status.

Response (200)

```
{
  "status": "ok",
  "version": "1.0.0",
  "name": "OmniLCS",
  "diameter_peers": [
    {
      "host": "dra01.epc.mnc380.mcc313.3gppnetwork.org",
      "realm": "epc.mnc380.mcc313.3gppnetwork.org",
      "state": "Connected",
      "transport": "sctp"
    }
  ],
  "active_sessions": 2,
  "completed_sessions": 47,
  "cells_loaded": 128,
  "cell_sync": {
    "last_sync": "2025-01-15T10:30:00Z",
    "last_result": "ok (128 cells)",
    "sync_count": 42
  },
  "uptime_seconds": 86400
}
```

Field	Type	Description
`version`	string	Application version
`name`	string	Instance name
`diameter_peers`	array	Connected Diameter peers with state
`active_sessions`	integer	Number of in-progress location sessions
`completed_sessions`	integer	Number of completed location sessions
`cells_loaded`	integer	Number of cells in the database
`cell_sync`	object	InfluxDB sync status
`uptime_seconds`	integer	Process uptime in seconds

Location Services

POST /api/location

A single, harmonised "get location" call. The caller supplies the IMSI (and optionally a desired `method`/`accuracy`); OmniLCS decides *which 3GPP LCS role and interface* to use to satisfy it, and reports which one answered in the `source` field.

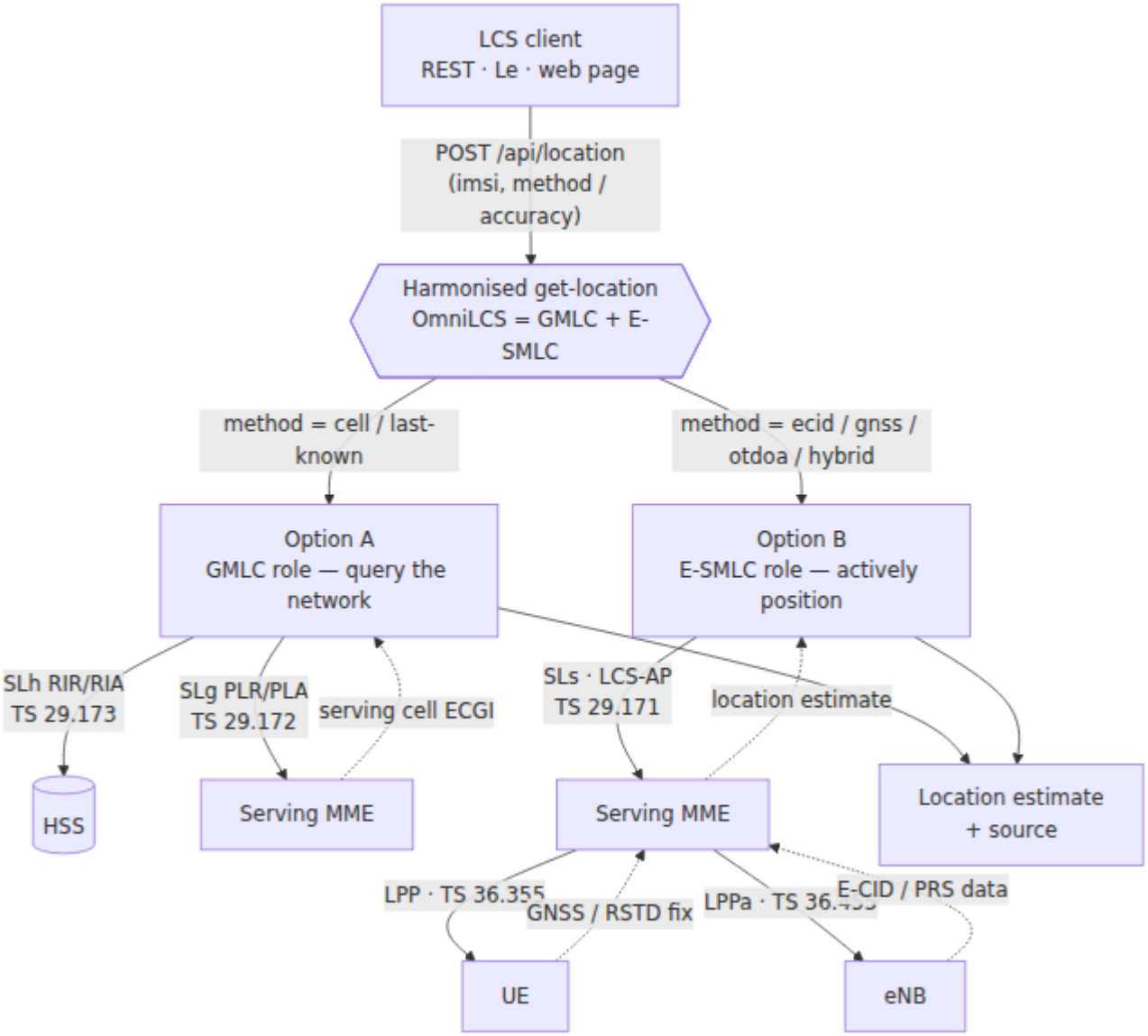
Every option is available here

OmniLCS is co-located as a **GMLC** and an **E-SMLC** (the LTE LCS network functions of TS 23.271) and also speaks **MAP/Lg** for legacy 2G/3G — so every standard way to locate a subscriber is reachable through this one endpoint:

Option	3GPP role	Interface(s)	How it locates	Best for
A — query the network (4G)	GMLC	SLh (TS 29.173) + SLg (TS 29.172)	No active positioning — the serving MME returns the UE's serving cell (ECGI) from its context	Fast, always-available coarse location; idle UEs (last-known)
B — actively position (4G)	E-SMLC	SLs / LCS-AP (TS 29.171), tunnelling LPP (TS 36.355) to the UE and LPPa (TS 36.455) to the eNB	OmniLCS runs E-CID / GNSS / OTDOA against the UE and radio	Precise fixes when the UE/radio support it
C — query the network (2G/3G)	GMLC	MAP / Lg over SS7 (TS 29.002): SRI-for-LCS to the HLR, then ProvideSubscriberLocation to the serving MSC/SGSN	The serving MSC/SGSN returns the serving cell (Cell-ID / TA class)	Subscribers camped on GERAN/UTRAI

Selection: for 4G, the requested `method`/`accuracy` chooses **A** (`cell`/last-known) or **B** (`ecid`/`gnss`/`otdoa`/`hybrid`). Setting `rat` to `2g`/`3g` chooses **C** regardless of method. In A/B the serving MME is always in the path (SLg

responder or SLs relay); in C the serving MSC/SGSN is. See 2G/3G location over MAP/Lg.



See Positioning Methods for how each method (Cell ID, E-CID, OTDOA, A-GNSS, hybrid) is obtained, their accuracy, and the accuracy→method auto-selection.

Request Body

```
{
  "imsi": "001010000000001",
  "method": "cell",
  "location_type": 1,
  "timeout": 30000,
  "mme_host": "mme01.epc.mnc380.mcc313.3gppnetwork.org"
}
```

Parameter	Type	Required	Default	Description
`imsi`	string	Yes	--	IMSI of the UE to locate
`method`	string	No	`cell`	Positioning method: `cell` `ecid`, `gnss`, `otdoa`, `hybrid`. Active methods (`ecid`/`gnss`/`otdoa`/`hybri` drive the E-SMLC over SL `cell` uses an SLg PLR.
`accuracy`	integer	No	--	Requested horizontal accuracy in metres. If set auto-selects the method (tighter → GNSS/OTDOA/E CID); for the cell/last-known path it is also carried to the MME as LC QoS.
`location_type`	integer	No	`1`	SLg-Location-Type (cell/la known path): `0` current, current or last known, `2` initial. The default `1` returns the last-known serving cell for idle UEs rather than failing a current-location request.
`mme_host`	string	No	--	Destination-Host (serving MME). If omitted, it is resolved via an SLh `LCS-Routing-Info-Request` to the HSS.
`rat`	string	No	`4g`	Radio access technology. `2g`/`3g` route over MAP/Lg

Parameter	Type	Required	Default	Description
				(Option C) to the serving MSC/SGSN; `4g` uses SLg/SLs (Options A/B) as selected by `method`.
`node_gt`	string	No	--	2G/3G only: serving MSC/SGSN global title. If omitted, it is resolved via MAP SendRoutingInfoForLCS t the HLR.
`timeout`	integer	No	`30000`	Timeout in milliseconds

> The requested `method` (or `accuracy`) selects *how* OmniLCS positions the UE: active methods run the OmniLCS E-SMLC (LPP/LPPa over SLs), while `cell`/last-known queries the serving MME over SLg. On the SLg path the MME ultimately decides what it can return (typically the serving cell ECGI).

Response (200)

```
{
  "status": "ok",
  "imsi": "001010000000001",
  "method": "cell",
  "source": "slg",
  "latitude": 40.7128,
  "longitude": -74.0060,
  "altitude": null,
  "uncertainty": 10.5,
  "confidence": null,
  "cell_id": 43559,
  "plmn": {"mcc": "001", "mnc": "01"},
  "ecgi": "00F1100000AA27",
  "serving_node": {"name": "mme01...", "realm": "epc..."},
  "duration_ms": 5230,
  "timestamp": "2025-01-15T10:30:00Z"
}
```


The response **merges every field the positioner returned** with the request envelope, so the exact set varies by path. `latitude`/`longitude` are decoded from the GAD `Location-Estimate`; `source` is `"slg"` for results from the serving MME over SLg, or the positioning method (`gnss`, `ecid`, …) for active E-SMLC fixes. Cell/SLg fixes additionally include the decoded ECGI (`cell_id` + `plmn`) alongside the raw `ecgi`. Binary AVPs (raw ECGI, GAD estimate, Serving-Node) are hex-encoded.

Field	Type	Description
`imsi`	string	UE identifier
`method`	string	Method requested
`source`	string	Actual positioning source used (`s` `ecid`, …)
`latitude`	float/null	Latitude in decimal degrees
`longitude`	float/null	Longitude in decimal degrees
`altitude`	float/null	Altitude in meters (if available)
`uncertainty`	float/null	Horizontal position uncertainty in
`uncertainty_altitude`	float/null	Vertical uncertainty in meters (w
`confidence`	integer/null	Confidence percentage
`shape`	string/null	GAD shape of the estimate (e.g. `ellipsoid_point_with_uncertai`
`cell_id`	integer/null	Decoded E-UTRAN cell-id (cell/SL
`plmn`	object/null	Decoded serving PLMN `{mcc, mn` path)
`ecgi`	string/null	Raw 7-byte ECGI, hex (cell/SLg p
`serving_node`	object/null	Serving MME identity (Name/Rea
`age_of_location_estimate`	integer/null	Age of the estimate in minutes (v present)
`accuracy_fulfilment`	string/null	Whether the requested accuracy

Field	Type	Description
`ecid_measurements`	object/null	E-CID radio measurements (`ecid`
`duration_ms`	integer	Time taken in milliseconds
`timestamp`	string	ISO 8601 timestamp

Error Responses

Status	Reason	Description
400	`"imsi is required"`	Missing IMSI parameter
404	`"User not found"`	IMSI unknown
404	`"User not connected"`	UE not attached to network
422	`"No MME host available for this subscriber"`	No MME to route the request
504	`"Positioning timed out"`	Positioning method did not complete in time
500	(varies)	Internal error

Examples

Option A — coarse location via the GMLC role (SLh + SLg). Asking for `cell` (or any last-known request) resolves the serving MME and returns its serving cell:

```
curl -sk -X POST https://localhost:8443/api/location \
  -H "Content-Type: application/json" \
  -d '{"imsi": "001010000000001", "method": "cell"}'
```

```
{
  "status": "ok",
  "imsi": "001010000000001",
  "method": "cell",
  "source": "slg",
  "latitude": -37.8183,
  "longitude": 144.9671,
  "altitude": null,
  "uncertainty": 1273.5,
  "confidence": null,
  "cell_id": 43559,
  "plmn": {"mcc": "001", "mnc": "01"},
  "ecgi": "00F1100000AA27",
  "serving_node": {"name":
"mme01.epc.mnc001.mcc001.3gppnetwork.org", "realm":
"epc.mnc001.mcc001.3gppnetwork.org"},
  "duration_ms": 412,
  "timestamp": "2026-06-22T01:14:55.218000Z"
}
```


Option B — precise location via the E-SMLC role (SLs / LPP). Asking for `gnss` (or supplying a tight `accuracy`) drives an active fix; `source` reflects the method that answered:

```
curl -sk -X POST https://localhost:8443/api/location \
  -H "Content-Type: application/json" \
  -d '{"imsi": "001010000000001", "method": "gnss", "timeout":
30000}'
```

```
{
  "status": "ok",
  "imsi": "001010000000001",
  "method": "gnss",
  "source": "gnss",
  "latitude": -37.8200,
  "longitude": 144.9834,
  "altitude": 558,
  "uncertainty": 9.5,
  "uncertainty_altitude": 1.1,
  "confidence": 90,
  "shape":
"ellipsoid_point_with_altitude_and_uncertainty_ellipsoid",
  "duration_ms": 10077,
  "timestamp": "2026-06-22T01:18:02.640000Z"
}
```


Equivalently, `-d '{"imsi": "001010000000001", "accuracy": 20}'` auto-selects A-GNSS (≤ 50 m → A-GNSS) and returns the same shape of response.

GET /api/location

List recent completed location fixes.

Query Parameters

Parameter	Type	Default	Description
`limit`	integer	`50`	Maximum number of results

Response (200)

```
{
  "status": "ok",
  "data": [
    {
      "imsi": "001010000000001",
      "method": "gnss",
      "state": "completed",
      "latitude": 40.7128,
      "longitude": -74.0060,
      "uncertainty": 10.5,
      "source": "gnss",
      "created_at": "2025-01-15T10:29:55Z",
      "completed_at": "2025-01-15T10:30:00Z"
    }
  ],
  "count": 1
}
```

GET /api/location/:imsi

Get the last known location for a UE by IMSI.

Path Parameters

Parameter	Type	Description
`imsi`	string	IMSI of the UE

Response (200)

```
{
  "status": "ok",
  "imsi": "001010000000001",
  "latitude": 40.7128,
  "longitude": -74.0060,
  "altitude": null,
  "uncertainty": 10.5,
  "confidence": null,
  "source": "gnss",
  "timestamp": "2025-01-15T10:30:00Z"
}
```


Error Responses

Status	Reason
404	`"No location found for IMSI"`
404	`"No completed location for IMSI"`

GET /api/location/:imsi/history

Get location fix history for an IMSI.

Path Parameters

Parameter	Type	Description
`imsi`	string	IMSI of the UE

Query Parameters

Parameter	Type	Default	Description
`from`	string	--	Start of time range (ISO 8601 or date)
`to`	string	--	End of time range (ISO 8601 or date)
`limit`	integer	`100`	Maximum number of results

Response (200)


```
{
  "status": "ok",
  "data": [
    {
      "timestamp": "2025-01-15T10:30:00Z",
      "imsi": "001010000000001",
      "method": "gnss",
      "latitude": 40.7128,
      "longitude": -74.0060,
      "altitude": null,
      "uncertainty": 10.5,
      "confidence": null,
      "source": "gnss",
      "duration_ms": 5230
    }
  ],
  "count": 1
}
```

GET /api/location/:imsi/history/csv

Export location history for an IMSI as a CSV file.

Path Parameters

Parameter	Type	Description
`imsi`	string	IMSI of the UE

Query Parameters

Same as `/api/location/:imsi/history`.

Response (200)

Returns a CSV file download with headers:

```
timestamp,imsi,method,latitude,longitude,altitude,uncertainty,confide
```

Content-Type: `text/csv` Content-Disposition: `attachment; filename="location_history_<imsi>_<date>.csv"`

Cell Database

GET /api/cells

List all cells in the database.

Response (200)

```
{
  "status": "ok",
  "data": [
    {
      "cell_id": "001-01-0001-01",
      "latitude": 40.7128,
      "longitude": -74.0060,
      "pci": 100,
      "earfcn": 1300,
      "radius": 500,
      "azimuth": null,
      "height": null,
      "prs_config": null,
      "updated_at": "2025-01-15T10:00:00Z"
    }
  ],
  "count": 1
}
```

GET /api/cells/:id

Get a single cell by cell_id.

Path Parameters

Parameter	Type	Description
`id`	string	Cell identifier

Response (200)

```
{
  "status": "ok",
  "data": {
    "cell_id": "001-01-0001-01",
    "latitude": 40.7128,
    "longitude": -74.0060,
    "pci": 100,
    "earfcn": 1300,
    "radius": 500,
    "azimuth": null,
    "height": null,
    "prs_config": {
      "bandwidth": 50,
      "config_index": 0,
      "num_dl_frames": 1,
      "cp_length": null,
      "num_antenna_ports": null
    },
    "updated_at": "2025-01-15T10:00:00Z"
  }
}
```


Error Response

Status	Reason
404	`"Cell not found: <id>"`

POST /api/cells

Create a new cell.

Request Body

```
{
  "cell_id": "001-01-0001-01",
  "latitude": 40.7128,
  "longitude": -74.0060,
  "pci": 100,
  "earfcn": 1300,
  "radius": 500,
  "azimuth": 120.0,
  "height": 30.0,
  "prs_config": {
    "bandwidth": 50,
    "config_index": 0,
    "num_dl_frames": 1,
    "cp_length": "normal",
    "num_antenna_ports": 2
  }
}
```

Parameter	Type	Required	Default	Description
`cell_id`	string	Yes	--	Unique cell identifier
`latitude`	float	Yes	--	Cell latitude (-90 to 90)
`longitude`	float	Yes	--	Cell longitude (-180 to 180)
`pci`	integer	No	--	Physical Cell Identity (0-503)
`earfcn`	integer	No	--	E-UTRA Absolute Radio Frequency Channel Number
`radius`	integer	No	`1000`	Coverage radius in meters
`azimuth`	float	No	--	Antenna azimuth in degrees
`height`	float	No	--	Antenna height in meters
`prs_config`	object	No	--	PRS configuration for OTDOA
`tac`	integer	No	--	Tracking Area Code (used for 4G CAP alert broadcast targeting)
`lac`	integer	No	--	Location Area Code (used for 2G and 3G CAP alert broadcast targeting)
`rat`	string	No	--	Radio Access Technology: `"4g"`, `"3g"`, or `"2g"`

Response (201)

Returns the created cell in the same format as `GET /api/cells/:id`.

Error Responses

Status	Reason
400	`"cell_id is required"`
400	`"latitude and longitude are required"`

PUT /api/cells/:id

Update an existing cell. Only provided fields are updated.

Path Parameters

Parameter	Type	Description
`id`	string	Cell identifier

Request Body

Any cell fields to update (same as POST, but all fields are optional).

Response (200)

Returns the updated cell.

Error Response

Status	Reason
404	`"Cell not found: <id>"`

DELETE /api/cells/:id

Delete a cell from the database.

Path Parameters

Parameter	Type	Description
`id`	string	Cell identifier

Response (204)

Empty body on success.

Error Response

Status	Reason
404	`"Cell not found: <id>"`

GET /api/cells/nearby

Find cells near a geographic point.

Query Parameters

Parameter	Type	Required	Default	Description
`lat`	float	Yes	--	Latitude of search center
`lon`	float	Yes	--	Longitude of search center
`radius`	float	No	`10`	Search radius in **kilometers**

Response (200)

```
{
  "status": "ok",
  "data": [
    {
      "cell_id": "001-01-0001-01",
      "latitude": 40.7128,
      "longitude": -74.0060,
      "pci": 100,
      "earfcn": 1300,
      "distance_km": 0.523
    }
  ],
  "count": 1
}
```


Results are sorted by distance (nearest first). Each entry includes a `distance_km` field.

Error Response

Status	Reason
400	`"lat and lon query parameters are required"`

POST /api/cells/sync

Trigger an immediate InfluxDB cell sync.

Request Body

None required.

Response (200)

```
{
  "status": "ok",
  "cells_synced": 128
}
```

Error Responses

Status	Reason
500	`"Sync failed: <reason>"`
503	`"Cell sync service unavailable"`

Deferred Location (GMLC)

Manage periodic and geo-fence triggered location sessions. See the GMLC & Le Interface guide for full details on session types and Diameter integration.

GET /api/deferred_location

List all active deferred location sessions.

Response (200):

```
{
  "status": "ok",
  "count": 1,
  "data": [
    {
      "session_id": "a1b2c3d4-e5f6-...",
      "type": "periodic",
      "imsi": "001010000000001",
      "method": "cell",
      "client_name": "rest-api",
      "status": "active",
      "interval_ms": 60000,
      "remaining_reports": 7,
      "total_reports": 10,
      "started_at": "2026-04-09T10:00:00Z",
      "last_fix_at": "2026-04-09T10:03:00Z"
    }
  ]
}
```


POST /api/deferred_location

Create a new deferred location session.

Periodic session request body:


```
{
  "type": "periodic",
  "imsi": "001010000000001",
  "method": "cell",
  "interval_seconds": 60,
  "count": 10
}
```

Field	Type	Required	Description
`type`	string	Yes	`"periodic"`
`imsi`	string	Yes	Subscriber IMSI
`method`	string	No	Positioning method: `cell`, `ecid`, `gnss`, `otdoa`. Default: `cell`
`interval_seconds`	integer	Yes	Seconds between fixes
`count`	integer	Yes	Total number of fixes to perform

Triggered session request body:

```
{
  "type": "triggered",
  "imsi": "001010000000001",
  "method": "cell",
  "event_type": "entering",
  "poll_interval_seconds": 30,
  "max_reports": 0,
  "areas": [
    {
      "type": "circle",
      "center": {"lat": -33.8688, "lon": 151.2093},
      "radius_meters": 500
    }
  ]
}
```

Field	Type	Required	Description
`type`	string	Yes	`"triggered"`
`imsi`	string	Yes	Subscriber IMSI
`method`	string	No	Positioning method. Default: `cell`
`event_type`	string	Yes	`"entering"`, `"leaving"`, or `"being_inside"`
`poll_interval_seconds`	integer	No	Seconds between position polls. Default: 30
`max_reports`	integer	No	Max trigger reports. `0` = unlimited
`areas`	array	Yes	List of area definitions (circle or polygon)

Response (201):

```
{"status": "ok", "message": "Periodic session created"}
```

GET /api/deferred_location/:session_id

Get the status of a deferred session.

Response (200):

```
{
  "status": "ok",
  "data": {
    "session_id": "a1b2c3d4-...",
    "type": "periodic",
    "imsi": "001010000000001",
    "status": "active",
    "remaining_reports": 7,
    "total_reports": 10
  }
}
```


DELETE /api/deferred_location/:session_id

Cancel an active deferred session.

Response (200):


```
{"status": "ok", "message": "Session cancelled"}
```


Status	Error
400	Missing or invalid parameters
404	Session not found

CAP Alerts

POST /api/cap

Submit a CAP XML alert for processing. The alert is parsed, polygon warning areas are resolved to TACs/LACs via the cell database, and the alert is either queued for approval or auto-broadcast depending on configuration.

Request Body

```
{
  "xml": "<alert
xmlns=\"urn:oasis:names:tc:emergency:cap:1.2\">...</alert>"
}
```

Parameter	Type	Required	Description
`xml`	string	Yes	Complete CAP v1.2 XML alert document

Response (201)

```
{
  "status": "ok",
  "data": {
    "id": "a1b2c3d4-e5f6-...",
    "status": "pending",
    "source": "http_post",
    "received_at": "2025-01-15T10:30:00Z",
    "matched_cells": 42,
    "tacs": [100, 101],
    "lacs": [5001],
    "mcc": "001",
    "mnc": "01",
    "broadcast_params": {
      "message_id": 4370,
      "repetition_period": 30,
      "num_broadcasts": 10,
      "message_text": "Tornado Warning...",
      "event": "Tornado Warning",
      "severity": "Extreme",
      "urgency": "Immediate"
    }
  }
}
```

The `status` field is `"pending"` when `require_approval` is `true`, or `"sent"` when auto-approved.

Error Responses

Status	Reason
400	`"xml field is required"`
422	Parse error details

GET /api/cap

List all alerts across all states (pending, active, history).

Response (200)

```
{
  "status": "ok",
  "data": {
    "pending": [...],
    "active": [...],
    "history": [...]
  }
}
```

GET /api/cap/:id

Get a single alert by ID.

Path Parameters

Parameter	Type	Description
`id`	string	Alert UUID

Response (200)

Returns the alert object.

Error Response

Status	Reason
404	`"Alert not found: <id>"`

PUT /api/cap/:id

Approve or reject a pending alert.

Request Body

```
{
  "action": "approve",
  "operator": "operator1"
}
```

Parameter	Type	Required	Description
`action`	string	Yes	`"approve"` or `"reject"`
`operator`	string	No	Operator name for audit trail (defaults to `"unknown"`)

Response (200)

Returns the updated alert object.

Error Responses

Status	Reason
400	`"action must be 'approve' or 'reject'"`
404	`"Alert not found: <id>"`

Error Response Format

All error responses follow this structure:

```
{
  "status": "error",
  "reason": "Human-readable error description"
}
```

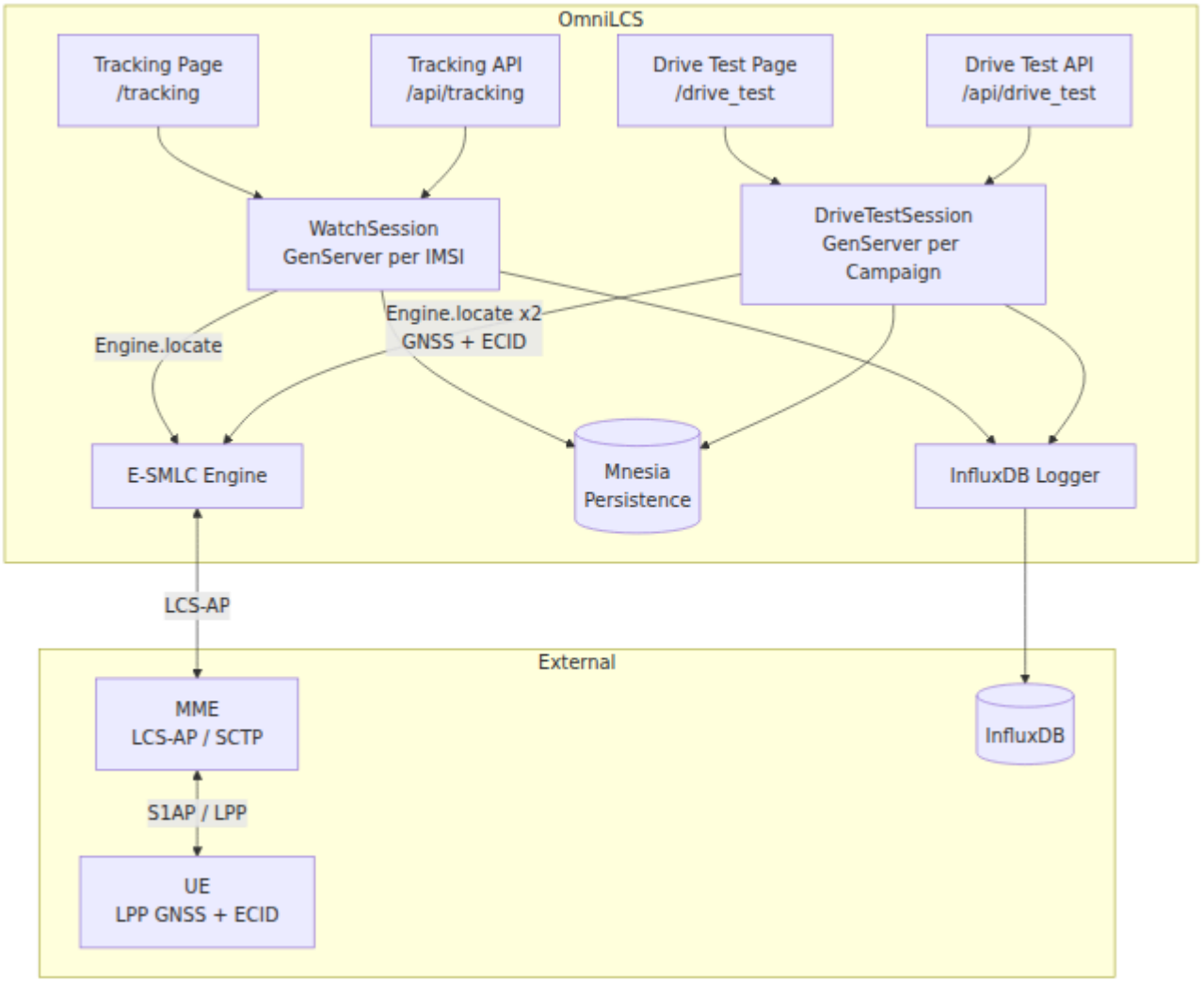
Common HTTP status codes:

Code	Meaning
200	Success
201	Created (cell creation)
204	No Content (cell deletion)
400	Bad Request (missing or invalid parameters)
404	Not Found (unknown resource)
422	Unprocessable Entity (valid request but cannot be fulfilled)
500	Internal Server Error
503	Service Unavailable
504	Gateway Timeout (positioning timeout)

Subscriber Tracking & Virtual Drive Test

OmniLCS provides two periodic location monitoring features: **Subscriber Tracking** for single-UE continuous monitoring, and **Virtual Drive Test** for multi-UE campaigns with combined GPS positioning and signal quality measurements.

Architecture



Subscriber Tracking

Subscriber Tracking periodically locates a single UE using a selected positioning method. Results are persisted in Mnesia and available via the web UI and REST API.

Web UI

Navigate to **Subscriber Tracking** in the sidebar. Enter an IMSI, polling interval (minimum 5 seconds), and select a positioning method (Cell, E-CID, GNSS, OTDOA). Click **Start Watching**.

Active watches are displayed in a table showing IMSI, method, interval, fix count, device status (Online/Offline/Error), last fix time, and last result. Click a row to view the fix history. Use the **Export CSV** or **Export KML** buttons to download data.

REST API

Method	Path	Description
`GET`	`/api/tracking`	List active tracking sessions
`POST`	`/api/tracking`	Start tracking. Body: `{"imsi": "...", "method": "gnss", "interval": 30}`
`GET`	`/api/tracking/:imsi`	Get tracking history for an IMSI
`DELETE`	`/api/tracking/:imsi`	Stop tracking an IMSI
`GET`	`/api/tracking/:imsi/export/csv`	Download CSV export
`GET`	`/api/tracking/:imsi/export/kml`	Download KML export

Device Status

The tracking session handles device online/offline transitions:

Status	Condition	Behavior
Online	`Engine.locate` succeeds	Fix stored with coordinates
Offline	No connected MME or UE unreachable	Error recorded, session continues polling
Error	Locate failed for other reasons	Error recorded, session continues polling

The session never stops on errors. It continues polling at the configured interval until explicitly cancelled.

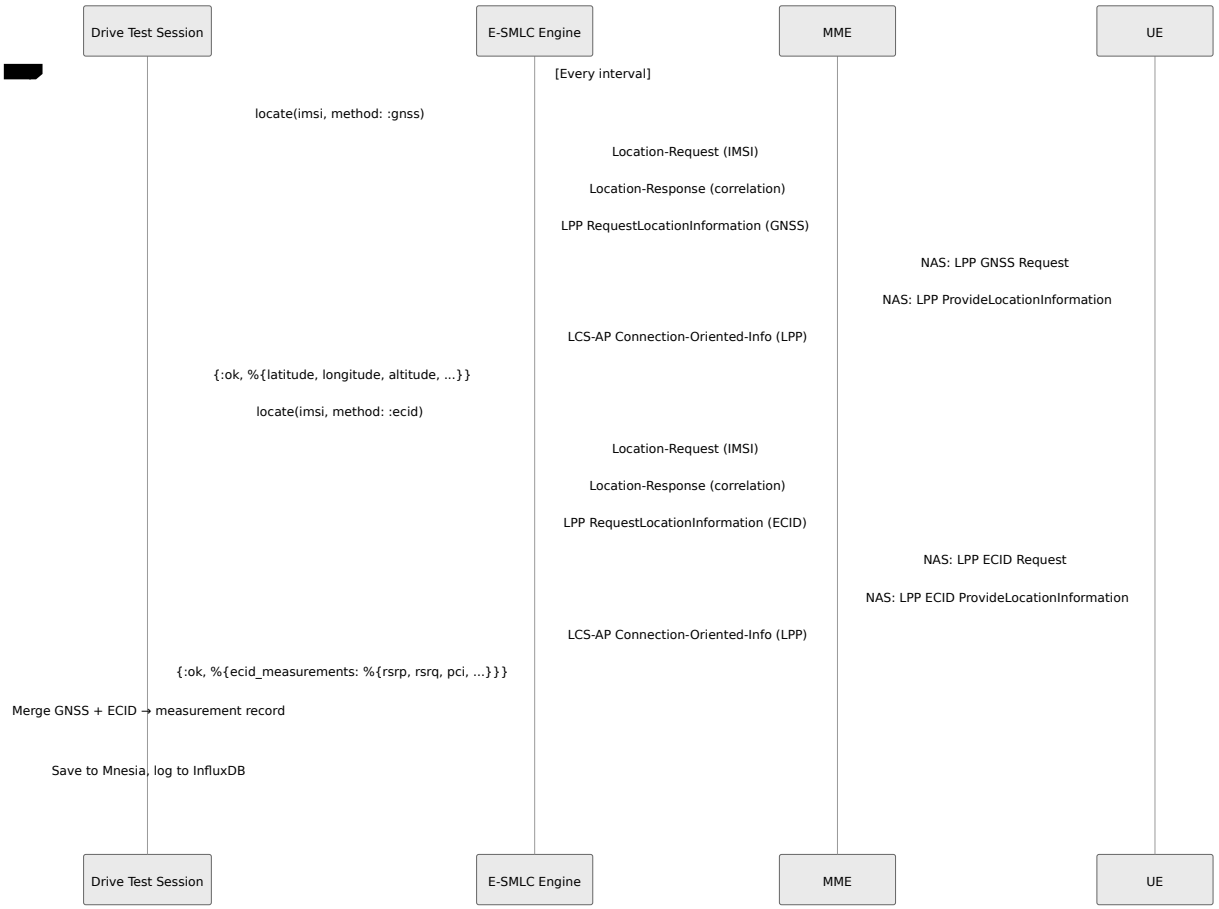
Virtual Drive Test

Virtual Drive Test runs multi-IMSI campaigns that combine **GNSS positioning** (for GPS coordinates) with **E-CID measurements** (for RSRP, RSRQ, serving cell, PCI, EARFCN). This is the signal quality measurement equivalent of a physical drive test.

How It Works

On each tick, the campaign:

1. For each IMSI in the campaign (up to 4 concurrent):
    - Sends an LPP **GNSS Request** to the UE via the MME for GPS coordinates
    - Sends an LPP **E-CID Request** to the UE for signal measurements
    - Merges the results into a single measurement record
2. Stores each measurement in Mnesia
3. Logs to InfluxDB (if configured)
4. Broadcasts results to the web UI via PubSub



Web UI

Navigate to **Virtual Drive Test** in the sidebar.

1. Enter a **Campaign Name** (e.g., "CBD Coverage Test")
2. Enter **IMSIs** (one per line or comma-separated)
3. Set the **Interval** (minimum 10 seconds)
4. Click **Start Campaign**

The campaigns table shows name, UE count, interval, tick count, and last tick time. Click a campaign row to view measurements. The measurement table shows per-fix data:

Column	Description
Time	Fix timestamp
IMSI	Target UE
Lat/Lon	GPS coordinates from GNSS
PCI	Physical Cell ID from E-CID
RSRP	Reference Signal Received Power (UE-reported)
RSRQ	Reference Signal Received Quality (UE-reported)
GNSS	GNSS status (ok/error)
ECID	E-CID status (ok/error)

RSRP values are color-coded: green (strong), yellow (medium), red (weak).

REST API

Method	Path	Description
`GET`	`/api/drive_test`	List active campaigns
`POST`	`/api/drive_test`	Start campaign. Body: `{"name": "...", "imsis": ["..."], "interval": 30}`
`GET`	`/api/drive_test/:id`	Get campaign measurements. Query: `?limit=200&imsi=filter`
`DELETE`	`/api/drive_test/:id`	Stop a campaign
`GET`	`/api/drive_test/:id/export/csv`	Download CSV with all measurements
`GET`	`/api/drive_test/:id/export/kml`	Download KML with signal-quality color coding

Example API Usage

Start a campaign:

```
curl -sk -X POST https://omnilcs:8445/api/drive_test \
  -H "Content-Type: application/json" \
  -d '{
    "name": "CBD Coverage Test",
    "imsis": ["001010000000001", "001010000000002"],
    "interval": 30
  }'
```

Get measurements:

```
curl -sk https://omnilcs:8445/api/drive_test/<campaign_id>?
limit=100
```

Export KML:

```
curl -sk
https://omnilcs:8445/api/drive_test/<campaign_id>/export/kml -o
coverage.kml
```

E-CID Measurements

The E-CID method requests RSRP, RSRQ, and UE Rx/Tx timing difference directly from the UE via LPP (3GPP TS 36.355). This approach works regardless of eNB LPPa support since the UE performs the measurements locally.

Measurement Fields

Field	Description	Source
`rsrp`	Reference Signal Received Power	UE measurement (mapped value, ~-44 to -140 dBm)
`rsrq`	Reference Signal Received Quality	UE measurement (mapped value, ~-3 to -19.5 dB)
`pci`	Physical Cell ID	Serving cell identifier
`earfcn`	E-UTRAN Absolute Radio Frequency Channel Number	Serving cell frequency
`cell_global_id`	Cell Global Identity (PLMN + Cell ID)	Serving cell
`ue_rx_tx_time_diff`	UE Rx-Tx time difference	Propagation delay estimate

API Response Example


```
{
  "status": "ok",
  "method": "ecid",
  "imsi": "001010000000001",
  "ecid_measurements": {
    "measurements": [
      {
        "pci": 373,
        "earfcn": 1825,
        "cell_global_id": {
          "cell_id": 4000,
          "plmn": {"mcc": "001", "mnc": "01"}
        },
        "rsrp": 40,
        "rsrq": 25,
        "ue_rx_tx_time_diff": 19
      }
    ]
  }
}
```


Export Formats

CSV

Comma-separated values with header row. Includes all measurement columns: IMSI, timestamp, coordinates, signal measurements, and status fields. Suitable for import into spreadsheet tools or analysis scripts.

KML

Google Earth / Google Maps compatible. Each fix becomes a Placemark with:

- Coordinates (lat/lon/alt)
- Timestamp
- Description with IMSI, RSRP, RSRQ, PCI

- Color-coded icons by RSRP signal strength (green/yellow/red)
- Per-IMSI track LineString connecting all fixes

Open in Google Earth, Google Maps, QGIS, or any KML-compatible GIS tool.

InfluxDB Integration

Both Subscriber Tracking and Virtual Drive Test log measurements to InfluxDB when the InfluxDB connection is configured in `runtime.exs`.

Measurements Written

InfluxDB Measurement	Source	Fields
`subscriber_tracking`	Tracking sessions	latitude, longitude, altitude, device_status
`drive_test`	Drive test campaigns	latitude, longitude, altitude, rsrp, rsrq, serving_pci, uncertainty

Tags

Tag	Description
`imsi`	Target UE IMSI
`method`	Positioning method used
`campaign_id`	Drive test campaign identifier (drive test only)

Persistence

All tracking and drive test data is stored in Mnesia disc_copies tables, surviving application restarts.

Mnesia Table	Key	Content
`:mnesia_tracking_history`	`{imsi, monotonic_time}`	Individual tracking fixes
`:mnesia_tracking_config`	`imsi`	Active tracking session configuration
`:mnesia_drive_test_measurements`	`{campaign_id, imsi, monotonic_time}`	Drive test measurements
`:mnesia_drive_test_config`	`campaign_id`	Active campaign configuration

History is automatically trimmed to 1,000 entries per IMSI (tracking) or 10,000 entries per campaign (drive test).

OmniLCS Web Interface Guide

The OmniLCS Control Panel is a real-time web interface served over HTTPS on port **443**. It is built with Phoenix LiveView, providing live-updating views without page reloads. All pages auto-refresh every 2-3 seconds.

Overview of Pages

The navigation bar presents pages in this order:

Page	Path	Description
Dashboard	`/dashboard`	System overview with key metrics
Location	`/location`	Location services testing interface
SLs Interface	`/sls`	SLs (LCS-AP) interface monitoring
Cell Databases	`/cells`	Cell database management and import
Cell Map	`/map`	Interactive cell map with area selection for broadcasts
Diameter	`/diameter`	Diameter peer monitoring
GMLC / Le	`/gmlc`	GMLC Le interface monitoring, deferred session management
Send Broadcast	`/send_broadcast`	Unified broadcast composition across 2G/3G/4G
Active Broadcasts	`/broadcasts`	Monitor and stop active broadcasts
CBC 2G	`/cbc`	2G CBSP monitoring
CBC 3G	`/cbc3g`	3G SABP monitoring
CBC 4G	`/cbc4g`	4G SBC-AP broadcast composer and monitoring
CAP Alerts	`/cap`	CAP alert ingestion, approval, and broadcast monitoring
Application	`/application`	OTP application resource viewer (built-in)

Page	Path	Description
Configuration	`/configuration`	Runtime configuration viewer (built-in)
Log	`/log`	Real-time log viewer (built-in)

Dashboard

Path: `/dashboard` **Refresh**: Every 2 seconds

The dashboard provides a high-level operational overview of the entire OmniLCS system.

Statistics Cards

Four summary cards across the top:

Card	Description
Active Sessions	Number of in-progress location sessions
Total Requests	Total location requests processed since startup
Success Rate	Percentage of completed vs total requests
Avg Response Time	Average positioning duration in milliseconds

Recent Location Requests

A table showing the 20 most recent location sessions with columns:

Column	Description
IMSI	UE identifier
Method	Positioning method (Cell, E-CID, GNSS, OTDOA, Hybrid)
MME	MME host that originated the request
Time	Request timestamp (HH:MM:SS)
Status	Color-coded badge (Active, Completed, Error, Timeout)

Connected Peers

List of SLs (LCS-AP) and Diameter peers in "Connected" state, showing:

- Peer hostname
- Interface type (SLs or Diameter)

System Status

Indicator lights for subsystem health:

Indicator	Green	Yellow	Red
SLs Interface (LCS-AP)	At least 1 MME SCTP association established	--	No MME connections
Diameter Service	At least 1 peer connected	--	No peers connected
Cell Database	Cells loaded	0 cells	--

Location

Path: `/location` **Refresh**: Every 2 seconds

Interactive location services testing interface for manually requesting UE positions.

Method Selector

Toggle buttons to select the positioning method:

Method	Description
Cell	Cell ID based, fastest, coarsest accuracy
E-CID	Enhanced Cell ID using eNB measurements via LPPa
GNSS	GPS/GNSS via LPP, highest accuracy
OTDOA	Multilateration from PRS measurements
Hybrid	Combined methods with fallback

Each method shows a description panel explaining how it works.

Request Form

- **IMSI Input**: Enter the IMSI of the UE to locate (e.g., `999990000000001`)
- **Request Location Button**: Sends the location request asynchronously

Requests run as background tasks. A spinning indicator shows while the request is in progress.

Session Counts

Summary badges showing:

- Total sessions
- Active sessions
- Completed sessions
- Error count
- Results received

Session History Table

All sessions sorted by creation time (newest first):

Column	Description
IMSI	UE identifier
Method	Positioning method
MME	Target MME host
Time	Session creation time
Duration	Time to complete (ms or seconds)
Status	Active, Completed, Error, Timeout

Completed sessions expand to show:

- **Coordinates**: Latitude and longitude
- **Uncertainty**: Position uncertainty in meters
- **Confidence**: Confidence percentage
- **Source**: Actual positioning source used
- **Map**: Embedded OpenStreetMap showing the position

Error sessions expand to show the error reason.

Notifications

Toast notifications appear for:

- Request initiated (info)
- Location retrieved (success, with coordinates)
- Location failed (error, with reason)

Notifications auto-dismiss after 5 seconds.

SLs Interface

Path: `/sls` **Refresh**: Every 2 seconds

Monitors the SLs (LCS-AP) interface between the E-SMLC and connected MMEs. Shows SCTP association status and connected MME peers.

Interface Status

Displays the LCS Application Protocol (LCS-AP) interface status:

- Protocol reference (3GPP TS 29.171)
- Listening configuration (PPID 29, port 9082)
- Number of connected MME peers
- Green/red indicator

Statistics

Card	Description
Connected MMEs	Number of active SCTP associations to MMEs
Total Associations	Total SCTP association count

MME Peers Panel

List of connected MME SCTP associations showing peer hostname and connection state.

Cell Database

Path: `/cells` **Refresh**: Every 2 seconds

Manages the cell position database used for Cell ID positioning and OTDOA calculations. All cell data is persisted to Mnesia and survives application restarts.

Header Controls

Control	Description
Cell count	Total number of cells in the database
Last sync	Timestamp and result of the last InfluxDB sync
Sync from InfluxDB	Trigger immediate InfluxDB cell sync
Map View / Table View	Toggle between table and map display
Import Cells	Open vendor-specific import modal
Add Cell	Open manual cell entry form

Search

Search cells by Cell ID, PCI, EARFCN, cell name, or source.

Table View

Columns displayed:

Column	Description
Cell ID	Unique cell identifier
PCI	Physical Cell Identity (LTE/NR) or Primary Scrambling Code (UMTS)
EARFCN	Radio frequency channel number
Latitude	Cell latitude in decimal degrees
Longitude	Cell longitude in decimal degrees
RAT	Radio Access Technology: GSM, UMTS, LTE, or NR
Source	Data source badge: **Huawei** (amber, XLSX import), **InfluxDB** (green, sync), or **Manual** (blue). Import timestamp shown below the badge.
Actions	Edit and Delete buttons

Import Cells

Clicking **Import Cells** opens a modal with a two-step workflow:

Step 1: Vendor Selection

Select the NMS vendor for the cell data being imported:

Vendor	Import Format
Huawei	U2020 XLSX export or CSV
Nokia	NetAct RAN Export CSV
Ericsson	ENM WCDMA Export CSV
ZTE	UMS Cell Export CSV
Other / JSON	JSON array paste

Step 2: Upload

Each vendor presents the appropriate upload interface.

Huawei XLSX (recommended): Drag and drop or select one or more Huawei U2020 Cell Report `.xlsx` files. Supports multi-file upload — for example, upload both UMTS and LTE reports together. Files import automatically once the upload completes.

The RAT type (GSM, UMTS, LTE, or NR) is detected automatically from each file based on:

- Sheet names in the workbook (e.g., a sheet named "UMTS", "LTE", "GSM", "NR")
- Column headers in the planning data sheet (e.g., `PScrambCode` for UMTS, `DlEarfcn` for LTE, `NRPCI` for NR, `BCCH` for GSM)

The parser reads the **Sheet1** tab (planning data with coordinates) and extracts:

RAT	Key Fields Imported
GSM	Cell ID, Cell Name, LAC, BCCH, BSIC, Longitude, Latitude, Antenna Height, Azimuth
UMTS	Cell ID, Cell Name, LAC, SAC, RAC, UARFCN Downlink, Scrambling Code, Longitude, Latitude, Antenna Height, Azimuth
LTE	Cell ID, Cell Name, PCI, TAC, DL EARFCN, Longitude, Latitude, Antenna Height, Cell Radius, Bandwidth
NR	Cell ID, Cell Name, NR PCI, TAC, NR DL EARFCN, Longitude, Latitude, Antenna Height, Cell Radius, Bandwidth

CSV (all vendors): Upload a CSV export from the selected NMS. The first row must be column headers. Column mapping is automatic based on the selected vendor format.

JSON: Paste a JSON array of cell objects. Required field: `cell_id`. Optional: `pci`, `earfcn`, `latitude`, `longitude`, `lac`, `tac`, `rat`.

Duplicate Handling

Importing cells with the same Cell ID as existing entries overwrites the existing data. This means re-importing the same file is safe and will update any changed values.

Data Persistence

All cell data is stored in Mnesia with `disc_copies` storage. Cells persist across application restarts without needing to re-import or re-sync.

Add / Edit Cell Form

Form fields organized in a grid:

Field	Description
Cell ID	Unique identifier (disabled when editing)
PCI	Physical Cell Identity (0-503)
EARFCN	Radio frequency channel number
Latitude	Decimal degrees
Longitude	Decimal degrees
Antenna Ports	1, 2, or 4
PRS Bandwidth (RBs)	6, 15, 25, 50, 75, or 100
PRS Config Index	0-4095
CP Length	Normal or Extended

Delete Confirmation

Clicking "Del" on a cell row shows a confirmation modal before deletion.

Cell Map

Path: `/map`

Interactive map showing all cells from the cell database on a dark-themed CartoDB base map using Leaflet. Cells are plotted as colored markers based on their RAT type.

Marker Colors

RAT	Color
GSM (2G)	Orange
UMTS (3G)	Blue
LTE (4G)	Green
NR (5G)	Purple
Unknown	Grey
Selected	Amber

Markers are clustered at lower zoom levels using Leaflet MarkerCluster. Clicking a marker shows a popup with the cell's details: Cell ID, name, technology, LAC/SAC/TAC, PCI, EARFCN, and coordinates.

Area Selection for Broadcasts

Use the polygon or rectangle drawing tools in the top-right corner to select cells within a geographic area. When cells are selected:

- The selection bar at the bottom shows the number of selected cells and unique LAC(s), SAC(s), and TAC(s)
- A **Send Broadcast to Selected** button appears, linking to the Send Broadcast page with the LACs, SACs, and TACs pre-filled

The Send Broadcast form auto-selects the appropriate target networks based on the selected area codes:

- LACs present → 2G (CBSP) enabled
- SACs present → 3G (SABP) enabled
- TACs present → 4G (SBc-AP) enabled

Diameter

Path: `/diameter` **Refresh**: Every 2 seconds

Monitors Diameter peer connections for the remaining Diameter interfaces (SLg and others). The E-SMLC-to-MME SLs interface uses native LCS-AP over SCTP and is monitored separately on the Dashboard page.

Interface Status

Two panels showing interface health:

Interface	Description
SLg Interface (TS 29.172)	GMLC to MME via DRA, Application ID 16777264
Other Diameter Interfaces	Additional Diameter applications as configured

Each shows a green/red indicator dot and description.

Peer Summary

Connected and disconnected peer counts displayed at the top.

Peer Table

Column	Description
Peer	Diameter host identity
Realm	Diameter realm
IP Address	Transport address (protocol://ip:port)
Status	Connected or Disconnected badge

Clicking a peer row expands to show detailed information:

Detail	Description
Connection Initiation	Whether OmniLCS initiates the connection
Transport	SCTP or TCP
Product Name	Remote peer's Diameter Product-Name
Advertised Applications	Application IDs supported by the peer

CBC 2G

Path: `/cbc` **Refresh**: Every 2 seconds

Monitors the CBSP interface for 2G cell broadcast.

Interface Info

Shows the CBSP interface status:

- Protocol reference (3GPP TS 48.049)
- Listening port number
- Number of connected peers
- Green/red indicator

Statistics Cards

Card	Description
Connected BSCs	Number of peers with status `:connected`
Total Connections	Total peer count (including any non-connected)
Recent Messages	Count of CBSP messages in the recent message buffer

Connected BSCs Panel

List of connected CBSP peers showing:

- Peer IP address and port
- Connection ID
- Connected timestamp
- Connection status badge

Recent CBSP Messages Panel

Table of the 20 most recent CBSP messages:

Column	Description
Time	Message timestamp (HH:MM:SS)
Peer	Connection ID of the source peer
Message	Human-readable message type (e.g., "WRITE REPLACE COMPLETE")
Type	Color-coded badge (OK for complete, FAIL for failure, ERR for error, INFO for others)

Messages are received via PubSub subscriptions to `cbsp:connections` and `cbsp:messages` topics.

CBC 3G

Path: `/cbc3g` **Refresh**: Every 2 seconds

Monitors the SABP interface for 3G cell broadcast over the Iu-BC interface (3GPP TS 25.419).

Interface Info

Shows the SABP interface status:

- Protocol reference (3GPP TS 25.419)
- Listening port number
- Number of connected RNCs
- Green/red indicator

Statistics Cards

Card	Description
Connected RNCs	Number of RNC peers with status `:connected`
Total Connections	Total peer count (including any non-connected)
Recent Messages	Count of SABP messages in the recent message buffer

Connected RNCs Panel

List of connected SABP peers showing:

- Peer IP address and port
- Connection ID
- Connected timestamp
- Connection status badge

Recent SABP Messages Panel

Table of the 20 most recent SABP messages:

Column	Description
Time	Message timestamp (HH:MM:SS)
Peer	Connection ID of the source RNC
Message	Human-readable message type (e.g., "WRITE REPLACE COMPLETE", "RESTART INDICATION")
Type	Color-coded badge (OK for complete, FAIL for failure, ERR for error, INFO for others)

Messages are received via PubSub subscriptions to `sabp:connections` and `sabp:messages` topics.

Send Broadcast

Path: `/send_broadcast` **Refresh**: Live

Unified broadcast composition interface for sending emergency warnings or test messages simultaneously across 2G (CBSP), 3G (SABP), and 4G (SBc-AP) networks. Select one or more target networks and the broadcast is dispatched to all connected peers on each selected network.

The default Message ID is set to **4355 (ETWS Test)** for safety.

Pre-fill from Cell Map

When navigating from the Cell Map's area selection, the form is automatically populated:

- **Target Networks** are enabled based on the area parameters (LACs → 2G, SACs → 3G, TACs → 4G)
- **LAC, SAC, and TAC fields** are pre-filled with the values from the selected cells
- **Cell Scope** is set to "By LAC"

This allows a complete workflow: select cells geographically on the map, click "Send Broadcast to Selected", compose your message, and send.

Active Broadcasts

Path: `/broadcasts` **Refresh**: Every 3 seconds + real-time PubSub

Unified view of all active cell broadcasts across 2G (CBSP), 3G (SABP), and 4G (SBc-AP) networks in a single page. Broadcasts are tracked from the time they are sent until they are stopped or expire. The view auto-refreshes every 3 seconds and updates in real time when broadcast state changes.

Summary Cards

Card	Description
Total Active	Combined active broadcast count across all networks
2G (CBSP)	Active 2G broadcasts
3G (SABP)	Active 3G broadcasts
4G (SBC-AP)	Active 4G broadcasts

Use the Stop button on any broadcast row to cancel it on its respective network.

CBC 4G

Path: `/cbc4g` **Refresh**: Every 3 seconds

Full-featured 4G cell broadcast management interface for composing, sending, monitoring, and stopping SBC-AP broadcasts.

Statistics Cards

Card	Description
Connected MMEs	Number of established SBC-AP SCTP associations
Active Broadcasts	Number of currently active 4G broadcasts
Total Sent	Total broadcasts in history

Compose Broadcast Form

Field	Description
Message Text	Broadcast message content
Message ID	16-bit CB message identifier (e.g., 4370 for CMAS)
Serial Number	16-bit serial number
MCC	Mobile Country Code (e.g., "313")
MNC	Mobile Network Code (e.g., "380")
TACs	Comma-separated Tracking Area Codes
Warning Type	None, Earthquake, Tsunami, Earthquake + Tsunami, Test, Other
Repetition (sec)	Seconds between broadcast repetitions
Num Broadcasts	Total number of times to broadcast the message

Clicking "Send Broadcast" encodes the message (GSM 7-bit), constructs a Write-Replace-Warning-Request PDU, and sends it to all connected MMEs.

Connected MMEs Panel

List of SBC-AP SCTP associations:

- MME hostname
- IP address, port, and SCTP association ID
- Connection status badge (Connected / state name)

Active Broadcasts Panel

Table of currently active broadcasts:

Column	Description
MSG ID	Message identifier in hex (e.g., 0x1112)
SERIAL	Serial number in hex
MESSAGE	Broadcast text (truncated)
STATUS	Sent, Stopping, Stopped, Acknowledged
ACTION	Stop button (for sent/acknowledged broadcasts)

Clicking "Stop" sends a Stop-Warning-Request for the broadcast.

Broadcast History

Table of all broadcasts (last 100):

Column	Description
MSG ID	Message identifier in hex
SERIAL	Serial number in hex
MESSAGE	Broadcast text (truncated)
TIME	Sent timestamp (HH:MM:SS)
STATUS	Final status badge

State updates are received via PubSub subscriptions to `cbc:state` and `cbc:connections` topics.

CAP Alerts

Path: `/cap` **Refresh**: Every 3 seconds + real-time PubSub

Operator approval workflow for CAP (Common Alerting Protocol) alerts received from external alerting authorities. See the CAP Alert Operations Guide for full details on configuration, feed polling, and polygon resolution.

Statistics Cards

Card	Description
Pending Approval	Number of alerts awaiting operator action (amber highlight when > 0)
Active Broadcasts	Number of currently broadcasting alerts
Total Processed	Total alerts in history
Approval Mode	Current setting: "Manual" (require_approval: true) or "Auto"

Pending Alerts Panel

Only visible when `require_approval` is `true` in the CAP configuration.

Column	Description
TIME	When the alert was received
EVENT	Alert event type (e.g., "Tornado Warning")
SEVERITY	Alert severity (Extreme, Severe, Moderate, Minor)
CELLS	Number of cells matched by polygon resolution
STATUS	Pending badge (amber)
ACTIONS	Preview, Approve, Reject buttons

Preview expands to show the full alert description, source, matched TACs/LACs, message ID, and PLMN.

Approve triggers immediate broadcast via SBC-AP (4G), SABP (3G), and CBSP (2G).

Reject moves the alert to history as rejected.

Active Broadcasts Panel

Column	Description
EVENT	Alert event type
MSG ID	CB message identifier
TACs	Targeted Tracking Area Codes
STARTED	Broadcast start time
STATUS	Broadcasting (blue) or Sent (green)

Alert History Panel

Column	Description
TIME	Sent or received timestamp
EVENT	Alert event type
SEVERITY	Alert severity
CELLS	Matched cell count
TACs/LACs	Targeted area codes
STATUS	Sent (green), Rejected (red), or Failed (red)

A toast notification appears when a new pending alert is received.

GMLC / Le Interface

Path: `/gmlc` **Refresh**: Every 3 seconds + real-time PubSub

Monitoring and management for the GMLC Le interface and deferred location sessions. See the GMLC & Le Interface Operations Guide for full configuration and Diameter protocol details.

Statistics Cards

Card	Description
Active Sessions	Total active deferred location sessions
Periodic	Number of active periodic location sessions
Geo-fence	Number of active triggered/geo-fence sessions

Authorized LCS Clients Panel

Column	Description
NAME	Client identity (matched against Diameter AVPs)
TYPE	LCS client type (emergency_services, value_added_services, etc.)
ALLOWED METHODS	Positioning methods this client may request
RATE LIMIT	Requests per minute (or "Unlimited")

Active Deferred Sessions Panel

Column	Description
SESSION ID	Truncated session UUID
TYPE	Session type badge: periodic (green) or triggered (purple)
IMSI	Subscriber IMSI
METHOD	Positioning method
CLIENT	Name of the requesting LCS client
PROGRESS	Periodic: completed/total reports. Triggered: remaining reports or "Active (unlimited)"
LAST FIX	Time of the most recent position fix
ACTIONS	Cancel button to terminate the session

Session History Panel

Shows the 50 most recent completed or cancelled sessions with session ID, type, IMSI, status (completed/cancelled), and start time.

Built-in Pages

These pages are provided by the Control Panel framework.

Application (`/application`)

OTP application resource viewer showing:

- Running applications and their process trees
- Memory usage per process
- Message queue lengths

Configuration (`/configuration`)

Runtime configuration viewer displaying:

- All Application environment variables
- Grouped by application

Log (`/log`)

Real-time log viewer showing:

- Log messages from the `ControlPanel.Logger` backend
- Filterable by log level
- Auto-scrolling with pause capability