LOGZILLA DOCUMENTATION

Cisco Cucm

LogZilla App Store application: Cisco Cucm

LogZilla App Store · Generated April 27, 2026 · logzilla.ai/docs/logzilla-appstore/cisco-cucm

Overview

Cisco Unified Communications Manager (CUCM) generates Call Detail Records (CDR) and Call Management Records (CMR) that provide detailed information about call activity within an organization's telephony system. CDRs capture metadata about each call, including caller/callee numbers, call duration, timestamps, device information, and call routing details. CMRs provide quality metrics such as jitter, latency, and packet loss.

App Function

- Parse CDR and CMR JSON messages from the CDR Loader
- Extract metadata tags for filtering and analysis
- Categorize calls by duration, quality issues, and failure types
- Provide dashboards for call volume, quality metrics, and cluster analysis
- Alert on infrastructure failures, SIP errors, and quality degradation

Vendor Documentation

- Cisco Unified Communications Manager (CallManager) (https://www.cisco.com/c/en/us/support/unified-communications/unified-communications-manager-callmanager/series.html)
- Cisco CDR Field Descriptions (https://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cucm/callReportingBillingAdmin/14/cucm_b_reporting-billing-administration-guide-14/cucm_b_reporting-and-billing-administration-guide_chapter_01010.html)

Device Configuration

The CUCM app requires the **CDR Loader** tool to collect CDR/CMR files from CUCM and transmit them to LogZilla. See the CDR Loader Implementation Guide section below for deployment instructions.

Incoming Log Format

CUCM generates CDR and CMR files in CSV format. The CDR Loader processes these files and transmits each record to LogZilla as a JSON object via the HTTP ingest API.

Parsed Metadata Fields

The following tags are extracted from CDR and CMR records.

Extracted Tags

Tag Name	Example	Description
Vendor	Cisco	Vendor name
Product	CUCM	Product identifier
Event Class	voip	Cross-vendor classification
CDR Record Type	cdr	Type of record (cdr, cmr)
Calling Number	12063862614	Calling party phone number
Called Number	12062153407	Final called party phone number
SrcIP	::ffff:172.16.0.100	Originating endpoint IP address
DstIP	::ffff:10.0.0.42	Destination endpoint IP address
Cluster ID	c8801ccm-AMER-CL1	CUCM cluster identifier
Duration Category	Normal (30s-5m)	Call duration classification
Quality Issue	High Jitter	CMR quality threshold exceeded
Failure Category	Infrastructure	Call failure root cause
Orig Cause	Normal call clearing	Originating call termination cause
Orig Disposition	call successful	Originating call result
Dest Cause	User busy	Destination call termination cause
Dest Disposition	user busy	Destination call result

Triggers

Trigger	Description
`Cisco CUCM: Infrastructure Failure`	Circuit/bandwidth/network issues (notification enabled)
`Cisco CUCM: SIP Trunk Error`	SIP trunk server-side errors (notification enabled)
`Cisco CUCM: Call Quality Degradation`	CMR quality threshold exceeded
`Cisco CUCM: Zero Duration Call`	Immediate call failures
`Cisco CUCM: User Unavailable`	User busy/no answer/rejected

Dashboards

- **Cisco CUCM: Call Overview** - Call volume, duration distribution, failure categories, cluster breakdown, and live call stream
- **Cisco CUCM: Call Quality** - CMR quality metrics including jitter, latency, and packet loss analysis

Log Examples

CDR Record (JSON from CDR Loader)


```
{
  "cdrRecordType": "1",
  "globalCallID_callManagerId": "5",
  "globalCallID_callId": "8842845",
  "callingPartyNumber": "12063862614",
  "finalCalledPartyNumber": "12062153407",
  "origCause_value": "16",
  "destCause_value": "0",
  "duration": "245",
  "origDeviceName": "SEP74AD98E783AF",
  "destDeviceName": "SEP001122334455",
  "globalCallId_ClusterID": "c8801ccm-AMER-CL1",
  "origIpv4v6Addr": "::ffff:172.16.0.100",
  "destIpv4v6Addr": "::ffff:10.0.0.42"
}
```

CMR Record (JSON from CDR Loader)


```
{
  "cdrRecordType": "2",
  "globalCallID_callManagerId": "5",
  "globalCallID_callId": "8842846",
  "numberPacketsLost": "50",
  "jitter": "15",
  "latency": "45",
  "deviceName": "SEP74AD98E783AF",
  "globalCallId_ClusterID": "c8801ccm-AMER-CL1"
}
```


CDR Loader Implementation Guide

The CDR Loader processes CDR and CMR files generated by CUCM and transmits them to LogZilla for analysis. Deploy the CDR Loader using Docker Compose on a server with access to the CDR/CMR file directories.

Prerequisites

- Docker and Docker Compose installed on the deployment server
- File system access to CUCM CDR/CMR output directories
- Network connectivity to the LogZilla server
- LogZilla authentication token (see LogZilla API Documentation (https://www.logzilla.ai/docs/logzilla-api) for token management)

LogZilla App Installation

Install the Cisco CUCM app on the LogZilla server:

Navigate to **Settings > App Store**
Click **Add**
Select **Cisco CUCM**
Click **Install**

Shared vs. Separate Directory Configuration

CUCM deployments vary in how CDR and CMR files are stored:

- **Separate directories** - CDR and CMR files are written to different directories (most common)
- **Shared directory** - CDR and CMR files are written to the same directory

The Docker Compose configuration must match the deployment. Examples for both configurations are provided below.

Docker Compose Configuration (Separate Directories)

Create a `compose.yaml` file with the following content. Replace the paths, URL, and token with values for the target environment.

```
services:
  cdr-loader:
    container_name: cdr_loader
    image: logzilla/runtime:latest
    pull_policy: always
    volumes:
      - /path/to/cdr_files:/data/cdr
      - /path/to/cmr_files:/data/cmr
    command: >
      python3 /usr/lib/logzilla/bin/cdr-loader.py
      --cdr-directory /data/cdr
      --cmr-directory /data/cmr
      --logzilla-url http://logzilla.example.com/incoming
      --logzilla-token ingest-YOUR_TOKEN_HERE
      --max-batch-size 100
      --max-batch-time 5
      --cdr-delay 15
      --cmr-delay 5
      --call-delay 60
      --monitor-interval 10
```

Docker Compose Configuration (Shared Directory)

If CDR and CMR files are stored in the same directory, use this configuration:

```
services:
  cdr-loader:
    container_name: cdr_loader
    image: logzilla/runtime:latest
    pull_policy: always
    volumes:
      - /path/to/cdr_cmr_files:/data/cdr
    command: >
      python3 /usr/lib/logzilla/bin/cdr-loader.py
      --cdr-directory /data/cdr
      --cmr-directory /data/cdr
      --logzilla-url http://logzilla.example.com/incoming
      --logzilla-token ingest-YOUR_TOKEN_HERE
      --max-batch-size 100
      --max-batch-time 5
      --cdr-delay 15
      --cmr-delay 5
```

```
      --call-delay 60
      --monitor-interval 10
```

Required Parameters

Parameter	Description
`--logzilla-url`	LogZilla ingest URL (e.g., `http://logzilla.example.com/incoming`)
`--logzilla-token`	LogZilla authentication token

Optional Parameters

Parameter	Default	Description
`--delete-processed`	False	Remove CDR/CMR files after processing
`--max-batch-size`	100	Maximum events per batch
`--max-batch-time`	5	Maximum batch wait time (seconds)
`--cdr-delay`	15	Delay before processing CDR files (seconds)
`--cmr-delay`	5	Delay before processing CMR files (seconds)
`--call-delay`	60	Delay for call completion determination (seconds)
`--monitor-interval`	0	Queue monitoring interval (0 disables)
`--debug`	False	Enable debug logging

Container Management

Start the container:

```
docker compose up -d
```

Check container status:

```
docker compose ps
```

View logs:

```
docker compose logs -n 50 -f cdr-loader
```

Stop the container:

```
docker compose stop
```

Remove the container:

```
docker compose down
```

Troubleshooting

Add `--debug` to the command in `compose.yaml` to enable detailed logging for diagnostic purposes.