

Accounting and CDRs

Feature Description

Table of Contents

1 Introduction **1**

2 How CDRs Are Produced **2**
2.1 Accounting Events on the Call Processors . . . 2
2.2 CDR Collection on the Element Managers . . . 2
2.3 CDR Types . . . 3
2.4 Files, Rotation and Retention . . . 3

3 Default CDR Format **3**
3.1 SIP CDRs . . . 3
3.2 HTTP and ENUM CDRs . . . 4

4 Flex Accounting: Customising CDRs from the Service Logic **4**
4.1 Adding Custom Fields . . . 4
4.2 Selecting the CDR File . . . 5

5 CDR Post-Processing **5**
5.1 How It Works . . . 5
5.2 Enrichment and Rating . . . 6
5.3 Testing . . . 6

6 Monitoring **6**

7 Example: Customer ID and Rated Price for Billing **6**

1 Introduction

SRE produces Call Detail Records (CDRs) for the calls it handles, and optionally for the HTTP and ENUM requests processed by its service logics. CDRs are written as CSV files on the Element Managers, where they are available to billing, mediation and reporting systems. The SRE Product Description explains where accounting fits in the overall solution.

This document describes how CDRs are built from the events collected on the Call Processors, the default CDR format, how the service logic adds its own fields with Flex Accounting, how CDR post-processing builds tailored CDRs for enrichment and rating, and the online charging interface. It ends with a billing example.

2 How CDRs Are Produced

2.1 Accounting Events on the Call Processors

A Call Processor does not write CDRs itself. For each call, it records accounting events as they happen:

Event	When it is recorded
Start	The call is received, or a new routing attempt starts
Connect	The call is answered
Stop	An answered call is released
End	The call is rejected with a 4xx, 5xx or 6xx response and no further attempt is made
Recurse	A routing attempt fails and SRE tries another destination (crankback)
Redirect	SRE answers the call with a redirect (3xx) response
Refresh	The call is refreshed with an OPTIONS message

Events are first written to local files on the Call Processor, then forwarded to the CDR collector. If no collector is reachable, events remain on the Call Processor and are sent once a collector is available again. Each event is marked as sent only after the collector has acknowledged it, so no event is lost when an Element Manager is restarted.

2.2 CDR Collection on the Element Managers

The CDR collector runs on the Element Managers. It matches the events of each call and writes a CDR when the call reaches a state that requires one. For example, a successful call produces the events Start, Connect and Stop, and results in one Stop CDR. A call that is first rejected by one destination and then answered by a second one produces a Recurse CDR for the first attempt and a Stop CDR for the second.

Two collectors normally run, one on each Element Manager. Only the collector on the master Element Manager produces CDRs. The other one continuously copies the CDR files and the collector state (the calls in progress and their custom fields) from the master. If the Element Managers switch over, the new master continues with the calls already in progress, without losing their CDRs.

2.3 CDR Types

Each CDR type can be enabled or disabled in the system settings:

CDR type	Produced	Default
Start	When a call is received	Disabled
Stop	When an answered call ends	Enabled
End	When a call is rejected (4xx, 5xx, 6xx)	Enabled
Recurse	When SRE triggers a recursion or crankback	Enabled
Redirect	When SRE redirects a call (3xx)	Disabled
Interim	At a fixed interval during an answered call	Disabled
HTTP	For each HTTP request handled by a service logic	Disabled
ENUM	For each ENUM request	Disabled

Interim CDRs are enabled by setting an interval in seconds. They contain the information available at that time and let billing systems account for long calls before they end.

2.4 Files, Rotation and Retention

CDRs are written in CSV format, one record per line, in separate directories for SIP, HTTP and ENUM CDRs. The collector writes into a current file and closes it at a configurable rotation interval (5 minutes by default). File names start with `cdr-` followed by the start time of the period, so files sort in chronological order. A file is only given its final name once it is complete, which lets external systems pick up closed files only.

3 Default CDR Format

3.1 SIP CDRs

The default SIP CDR reserves 50 positions for fields defined by SRE. In the current release, the first 22 positions are used and positions 23 to 50 are left empty. Custom fields added by the service logic start at position 51 (see Flex Accounting).

The default fields fall into the following groups:

- **Timing**: setup, connect and disconnect times, each with the Call Processor that handled the event. The connect time is empty for unanswered calls.
- **Outcome**: the SIP status code (0 if there was no SIP error) and the CDR type (start, stop, end, recurse, interim or redirect), plus the redirect contact for redirect CDRs.
- **Call identification**: the SIP Call-ID and a counter identifying the routing attempt, incremented at each crankback.
- **Parties**: the From and To URIs, the initial Request URI and the last Request URI sent to the destination, each with its user part where relevant.
- **Network**: the source address and port of the INVITE, and the SRE address and port that received it.

The Accounting Guide lists every field with its position, and gives example records for successful, cancelled and rejected calls.

3.2 HTTP and ENUM CDRs

HTTP and ENUM CDRs have their own, shorter format:

- **HTTP**: timestamp, source IP address and port, URL, method, response code and request identifier.
- **ENUM**: timestamp, source IP address and port, queried number, ENUM zone, response code and request identifier.

They are also padded with empty positions, and custom fields are appended after them.

4 Flex Accounting: Customising CDRs from the Service Logic

With the SRE-LIC-FLEX-ACCOUNTING licence, the service logic can add its own fields to the CDR of each call. Any variable available in the service logic can be written: data from the SIP signalling, results of database queries in the data model, or answers from external systems.

4.1 Adding Custom Fields

Two nodes add custom fields, placed after the reserved positions:

- **Append CDR fields** appends one or more values after the custom fields already present. Each value is a variable or a fixed text.
- **Set CDR fields** writes values at given custom positions. Positions left unset remain empty. This keeps a fixed layout that does not depend on the path taken through the service logic.

Custom fields are linked to the routing attempt during which they were set. When a call is rerouted, the CDR of each attempt carries the fields set for that attempt.

The same nodes apply to HTTP and ENUM service logics, adding fields to HTTP and ENUM CDRs.

4.2 Selecting the CDR File

The **Set CDR filename** node sets a prefix for the file the CDR is written to, on a per-call basis. The CDR is then written to a file named `cdr-<prefix>-<timestamp>`. This can be used, for example, to separate the CDRs of different services, customers or partners into different files.

5 CDR Post-Processing

5.1 How It Works

CDR post-processing builds new, tailored CDRs from the default SIP CDRs. It runs on the Element Manager and processes each closed CDR file:

1. Each row of the file is passed to the service logic assigned to the CDR post-processing interface. The default fields are available as variables named after them, for example the setup, connect and disconnect times, the Call-ID or the From username.
2. The service logic can use any processing node: database queries in the data model, HTTP, SOAP, LDAP or Diameter queries to external systems, calculations and conditions.
3. The service logic ends with one of two output nodes. **Output CDR row** writes a new row containing exactly the columns defined in the node, in the order defined. **Skip CDR row** writes nothing for this record, for example to keep only outgoing calls or only answered calls.
4. The output rows are written to a separate post-processing directory. A **Set CDR filename** node in the post-processing service logic directs the output row to a file with that prefix, so a single input file can be split into several output files.

A file is processed as a whole: if a row fails, no output file is written for it and the input file is marked as failed, so a partial file never reaches billing. The error is logged with the row concerned. Tracing can also be enabled for CDR post-processing, based on the From and To usernames.

The post-processing nodes **Output CDR row** and **Skip CDR row** also require the SRE-LIC-FLEX-ACCOUNTING licence.

5.2 Enrichment and Rating

Typical uses of post-processing are:

- **Enrichment**: adding information that was not needed for routing, such as a customer account, a cost centre or a partner code, looked up in the data model or in an external system.
- **Rating**: computing a price from the call duration and its characteristics. The connect and disconnect times are available as date and time values, so the duration can be computed, and the rate looked up in a rate table of the data model.
- **Reformatting**: producing the exact column layout expected by a billing or mediation system.
- **Filtering**: dropping records that billing does not need.

Because post-processing runs on completed CDRs, after the call, lookups to external systems add no delay to call setup.

5.3 Testing

The Service Logic Editor has a **Simulate CDR** tab. A CDR row in CSV format is pasted as input and the post-processing service logic is executed on it, showing each step and the value of each variable. Simulations can be saved and run again after a change to the service logic.

6 Monitoring

- **Missing CDR alarm**: an alarm is raised if no CDR has been produced during a configurable interval.
- **Collector status**: each Call Processor reports whether it can reach the CDR collectors.
- **Open calls**: the number of calls in progress tracked by the collector is reported per Call Processor in the statistics.

7 Example: Customer ID and Rated Price for Billing

A service provider offers SIP trunking and wants its billing system to receive, for each answered call, the customer ID, the called number, the duration and a price. The data model contains a Customer table and a Rate table with a price per minute per destination prefix.

In the call routing service logic, once the customer has been identified from the incoming trunk, an **Append CDR fields** node adds the customer ID to the CDR. It becomes position 51 of each CDR for the call.

In the CDR post-processing service logic, for each CDR:

1. If the CDR type is not stop, or the connect time is empty, a **Skip CDR row** node drops the record.
2. A **Database query** retrieves the rate for the customer in `custom0` and the longest matching prefix of the called number.
3. An **Evaluate expression** node computes the duration in seconds from the connect and disconnect times, and a second one computes the price.
4. A **Set CDR filename** node with the prefix `billing` directs the row to the billing files.
5. An **Output CDR row** node writes the columns expected by the billing system: customer ID, calling number, called number, connect time, duration and price.

Every 5 minutes, when the collector closes a CDR file, post-processing produces a `cdr-billing-` file in the post-processing directory, which the billing system collects. The default CDRs remain available alongside for troubleshooting and reporting. When a new customer is provisioned, its CDRs carry its ID without any change to either service logic, and a rate change is a data update in the Rate table.