



Release Notes

SRE 4.0.1

Table of Contents

1	Introduction	2
2	What's New in SRE 4.0	2
2.1	Platform Upgrade	2
2.1.1	New Broker	2
2.1.2	New Licensing	3
2.2	Call Processing	3
2.2.1	SIP Registration Client	3
2.2.2	Hitless Upgrade	6
2.3	Service Logic Editor	6
2.3.1	Change Log	6
2.3.2	Tracing Local Storage	7
2.3.3	Service Logic Tagging	7
2.3.4	Bulk Service Logic Deletion	7
2.3.5	Service Selection Menu	8
2.3.6	Automatic Table Joins	8
2.3.7	Online Documentation	8
2.3.8	Improved Nodes	8
2.3.9	New Nodes	10
2.4	Built-in Scheduler	12
2.5	Statistics	13
2.5.1	Telegraf Integration	13
2.6	Alarms	14
2.7	Operations & Maintenance	14
2.7.1	Database Automatic Switchover	14
2.7.2	Improved Summary Log	15
2.7.3	Maintenance Jobs	16
2.7.4	Automatic Rolling Upgrade	16
2.8	REST API	17
2.8.1	Single Record Retrieval	17
2.8.2	Licenses Endpoint	17
2.9	Security & Auditing	18
2.9.1	Password Salting	18
2.9.2	Service Logic Fine-Grained Access Control	18
2.10	GUI	18
2.10.1	Basic/Advanced Settings Split	18

2.11	Miscellaneous Enhancements	19
3	Patch Versions Release Notes	19
3.1	Release 4.0.1	19
3.2	Release 4.0.2	20
4	Upgrade From 3.3	21
4.1	High-Level Procedures	22
4.1.1	Option 1A	22
4.1.2	Option 1B	22
4.1.3	Option 2A	23
4.1.4	Option 2B	23
4.2	Element Managers	24
4.2.1	SRE RPM Update	24
4.3	Call Processors	25
5	Downgrade From 4.0 to 3.3	26
6	Patch Upgrade Path From 4.0.x	26

1 Introduction

These release notes provide a comprehensive overview of the new features, enhanced functionalities, and resolved issues found in version 4.0 of SRE.

2 What's New in SRE 4.0

2.1 Platform Upgrade

All underlying frameworks and libraries have been updated to the latest versions. Starting from release 4, the reference Operating System is now RHEL 8, and support for RHEL 7 has been discontinued.

2.1.1 New Broker

The broker process, *sre-broker*, responsible for dispatching SIP requests from the SIP stack to the *sre-call-processor* instances, has been completely rewritten for improved performance.

2.1.2 New Licensing

The licensing system has been revised to provide greater flexibility. Several new licenses have been introduced to differentiate SIP redirect scenarios from SIP relay scenarios. Consequently, it is now possible to obtain different limits depending on the type of scenario. Additionally, a new license, based on the number of platform-wide simultaneous sessions, has been added.

Note

Please note that if you are updating from a previous release, we strongly recommend contacting Netaxis Support to request your licensing keys for the new licensing scheme to prevent any disruption of service.

2.2 Call Processing

2.2.1 SIP Registration Client

A new framework has been introduced, allowing the SRE to function as a SIP registration client for multiple endpoints. To facilitate this, a new process named *sre-remote-registration* has been implemented, responsible for periodically registering SIP endpoints. In its role as a SIP registration client, the SRE manages all aspects of SIP registration. This includes tasks like sending SIP REGISTER messages, handling authentications, and managing expiry timers. Configuration options for SIP endpoints are available through the new menu System -> SIP Remote Registration, or they can be dynamically generated from Datamodel data by creating a specific Service Logic (SL) to be activated. In this SL, the output node "Add registrations" can be utilized to provide a list of SIP endpoints (stored in a records list) for registration.

RECORD PROPERTIES

Username	<input type="text" value="tassadar"/>
Domain	<input type="text" value="netaxis.cloud"/>
Realm	<input type="text"/> <small>Leave empty to disable realm matching.</small>
Proxy	<input type="text" value="sip:10.0.10.182:5060"/> <small>Format: sip:host[:port] or sips:host[:port]</small>
Auth username	<input type="text" value="tassadar"/>
Auth password	<input type="password" value="*****"/>
Expires	<input type="text" value="300"/>

Figure 1: Registration endpoint record

New Node
×

Add registrations [?](#)

Node name *

Description

Records list *

 col

Registrations will be read from this records list

UUID column *

 col

UUID will be used in the REGISTER Contact header

Username column *

 col

Domain column *

 col

Realm column

 col

Leave empty to disable realm matching

Authentication username column *

 col

Authentication password column *

 col

Authentication proxy column *

 col

Expires column *

 col

Figure 2: Dynamic registration endpoints list

When deployed across multiple Call Processors (CPs), the *sre-remote-registration* processes will distribute the SIP endpoints among them for registration. In the event of a CP or process failure, the remaining elements will take over the role of SIP registration until the failed component is restored.

Note

To activate this feature, a new configuration definition is necessary, and you must use the following command to enable the WITH_REGISTER_CLIENT setting:

```
1 # echo '#!define WITH_REGISTER_CLIENT' >> /etc/kamailio/kamailio-local.cfg
```

and restart kamailio.

On the dashboard, a new tab titled “SIP Registration” has been added to facilitate the monitoring of SIP endpoints’ status.

2.2.2 Hitless Upgrade

SRE 4.0 enables the redirection of traffic from one call processor to another, offering a practical means to upgrade SRE with minimal disruption to ongoing traffic. For requirements please read the *Hitless upgrade* paragraph in the **Installation guide**.

To redirect traffic to another CP use the following command (confirmation is required):

```
1 # /opt/sre/bin/sre-admin kamailio set-redirection
```

A working CP is automatically chosen by this command and a pre-check is done to ensure th target CP is alive.

To deactivate traffic redirection, you can employ the following command:

```
1 # /opt/sre/bin/sre-admin kamailio reset-redirection
```

To get the status of traffic redirection run:

```
1 # /opt/sre/bin/sre-admin kamailio get-redirection
```

2.3 Service Logic Editor

2.3.1 Change Log

It is now possible to compare the current (development) version of a service logic with previous release versions of that same service logic. The tool is accessible from the Changelog tab of the Service Logic Editor (SLE). It lists changes in existing node parameters, added nodes, removed nodes, and modifications to links between nodes.

2.3.2 Tracing Local Storage

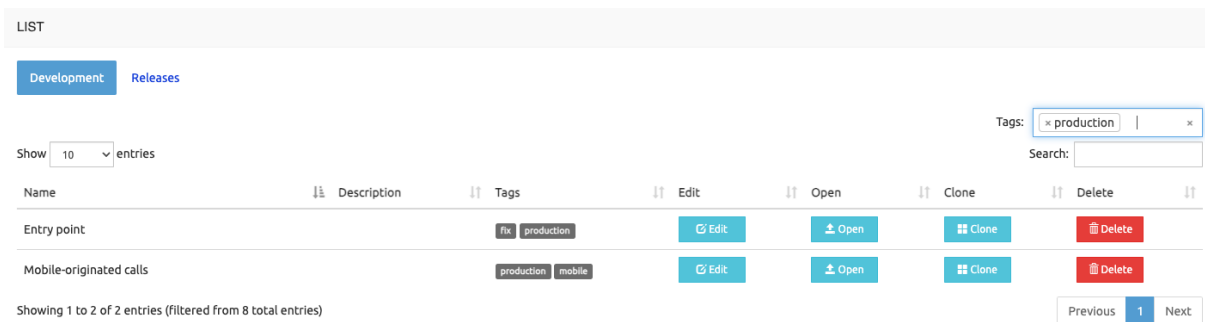
Tracing has been redesigned to store traces locally on both EMs rather than in-memory. This modification enables increased trace retention and preserves them in the event of an EM manager process restart.

2.3.3 Service Logic Tagging

A new tagging system has been implemented, enabling users to assign one or more dynamically created tags to service logics. This feature allows users to group service logics by category. These tags can also be automatically assigned upon importing a Service Logic. This functionality enables users to swiftly identify an imported service logic along with all its sub-service logics.



Figure 3: Tags assignment



Name	Description	Tags	Edit	Open	Clone	Delete
Entry point		fix production	Edit	Open	Clone	Delete
Mobile-originated calls		production mobile	Edit	Open	Clone	Delete

Figure 4: Tags filtering

2.3.4 Bulk Service Logic Deletion

A button has been added to allow the deletion of multiple service logics at once based on tags.

2.3.5 Service Selection Menu

New buttons have been added to the service selection menu, allowing immediate editing or opening of the currently active service logic.

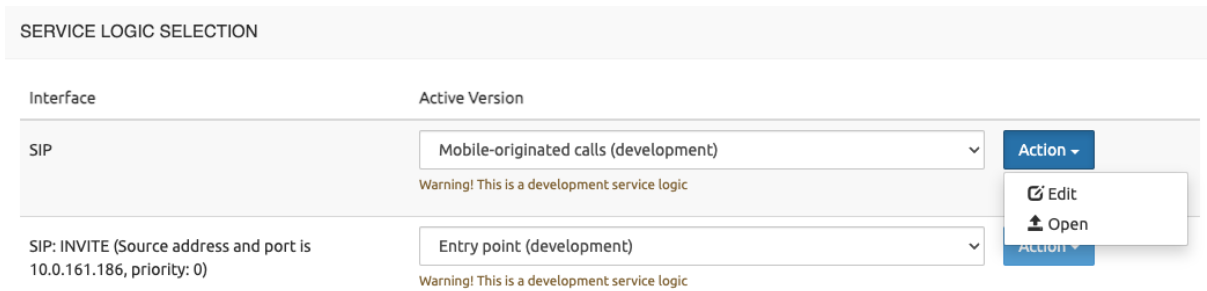


Figure 5: Service selection action menu

2.3.6 Automatic Table Joins

For the “Database Query” node, when executing joins between different tables, selecting the left column of the join now automatically populates the right column based on the foreign key of that column. This enhancement eliminates the need to manually select both sides of the join, reducing the potential for errors.



Figure 6: Automatic table joins

2.3.7 Online Documentation

Links have been incorporated to directly access online documentation from both the GUI menu and the nodes’ properties dialog.

2.3.8 Improved Nodes

- The “Regular Expression Match” node has been enhanced to support the use of variables for the regular expression, enabling dynamic behavior (e.g. retrieving the regular expression from

the database). This flexibility allows for the utilization of different regular expressions based on scenarios, allowing the matching of another variable with varying expressions.

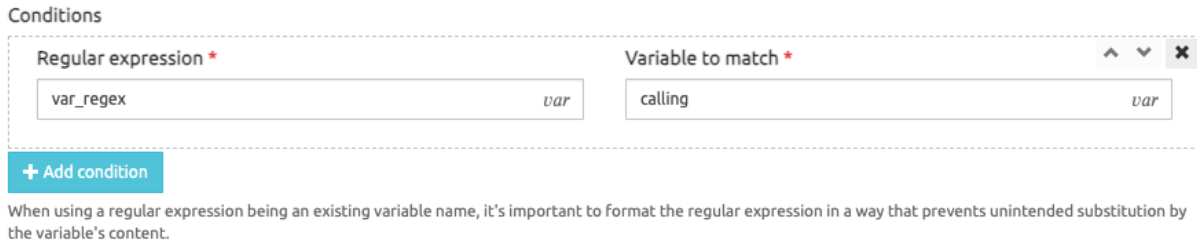


Figure 7: Variable regular expression

- For the HTTP query nodes, support for mutual TLS has been added. It is now possible to configure the client certificate, client private key, and CA certificate to establish mutual TLS with the external system.

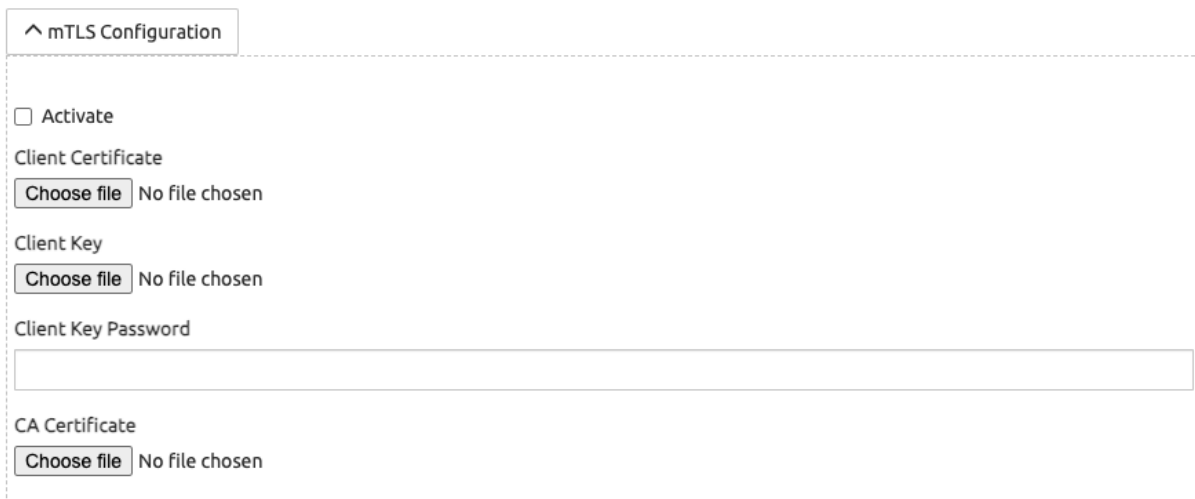


Figure 8: Mutual TLS parameters

- The “Set Variables” node has been enhanced to facilitate the indirect setting of a variable through a dereferencing-like mechanism. In practice, the target variable name is not static but is defined inside another variable.

Variables

Name *	Value *		
parameter_name	domain [var]	<input type="checkbox"/> If not set ?	<input checked="" type="checkbox"/> Indirect ?

[+ New variable](#)

By using indirect values, the target variable name or placeholder expression is read from the indirect value. Note that placeholder substitution can occur both in the indirect value expression and the final expression.

Figure 9: Indirect variable set

- The “DNS Query” node has been improved to support querying for SRV records, enabling the retrieval of priority, weight, port, and target information from SRV records.

Query type

SRV

For A queries, the output records list is populated with the column “address” to display the corresponding addresses. On the other hand, for SRV queries, the output records list includes the following columns: “priority,” “weight,” “port,” and “target.” These columns provide information about the priority, weight, port number, and target address of the SRV records, respectively.

Figure 10: DNS SRV query

- A new fire-and-forget option has been added to the “HTTP JSON Query” node, allowing the SRE to proceed to the next node without waiting for the result of the request.

Do not wait for result ?

Maximum pending requests ?

Figure 11: Fire-and-forget HTTP requests

2.3.9 New Nodes

- A new node, “Remove SIP Body,” has been introduced to facilitate the removal of the SIP body during response processing, such as in the case of a 180 Ringing or 183 Session Progress.
- Another node, “Strip To Tag,” has been added to remove the tag parameter from the To header during response processing.
- Two new nodes have been introduced to facilitate data caching through a MongoDB database. The “Set Cached Data” node enables the storage of data by specifying a key, value, and time-to-live. The corresponding “Get Cached Data” node facilitates the retrieval of cached data. Both nodes allow the definition of the MongoDB connection string, optionally incorporating replica set information. A built-in cooldown mechanism ensures a graceful bypass if the underlying

database becomes temporarily unavailable. Together, these nodes enable platform-wide data caching accessible from any Call Processor.

New Node
✕

Set cached data (MongoDB) [?](#)

Node name *

Description

MongoDB connection string *

```
mongodb://db1,mdb2,mdb3/?replicaSet=sre&connectTimeoutMS=100&socketTimeoutMS=100&w=1&readPreference=nearest&serverSelectionTimeo [var]
```

MongoDB connection string in format mongodb://<user>:<password>@<host-1>,...,<host-n>/?replicaSet=<replica-set>&connectTimeoutMS=100&socketTimeoutMS=100&w=1&readPreference=nearest&serverSelectionTimeoutMS=100. Some options are optional. Use appropriate timeout settings as this node may introduce delay in processing if MongoDB is unreachable/unhealthy.

Cooldown duration *

 [var]

If the database cannot be reached, no additional attempts will be made during this time.

Errors to trigger cooldown *

 [var]

Number of consecutive errors before cooldown procedure is initiated.

Key *

 [var]

Value *

 [var]

Time to live *

 var

Time to live in seconds

Figure 12: New platform-wide caching nodes

- A new node, “Set CDR Filename,” has been introduced to enable the selection of the CDR filename prefix used for the output CDR. This feature facilitates the directing of CDR generation to different CDR files on a per-call basis.

New Node ×

Set CDR filename [?](#)

Node name *	Description
<input type="text" value="Set mobile CDR filename"/>	<input type="text"/>

Prefix *

The prefix value will be utilized to name the output CDR file following the format: cdr-[prefix]-[timestamp].

Figure 13: Custom CDR filename prefix parameter

2.4 Built-in Scheduler

A new service logic interface has been introduced to execute Service Logic at regular intervals. The execution itself is managed by a new process called “sre-scheduler.” Each SL execution begins with a call descriptor containing two variables: “now,” filled with the current timestamp, and “hostname,” set to the hostname of the running SL (enabling distinct behavior based on the host). The execution interval is controlled by the custom endpoint mechanism, allowing the definition of various custom endpoints with different execution frequencies for the SL.

SIP
HTTP
Scheduled service logic

NEW CUSTOM ENDPOINT

Name

Interval

Unit

Create

CUSTOM ENDPOINTS

Show entries Search:

Name	Interval	Unit	Delete
every minute	60	second	Delete

Showing 1 to 1 of 1 entries

Figure 14: Custom schedulings

2.5 Statistics

2.5.1 Telegraf Integration

Telegraf integration has been implemented, allowing the Telegraf agent to transmit statistical data to the statistics database (operating on InfluxDB) on the Element Managers when activated. This data can subsequently be visualized through the built-in dashboard functionality of SRE. To support this integration, several built-in graph definitions have been introduced:

- Disk activity operations I/O
- Disk activity bytes I/O
- MongoDB commands
- MongoDB connections
- Processes
- CPU Usage over time
- DB cache
- DB connections
- Records activity
- Network bytes I/O

- Network packets I/O
- Network TCP segments I/O
- Network UDP datagrams I/O

These new metrics offer deeper insights into the health and performance of the platform, encompassing aspects such as the operating system, network, and various subsystems like PostgreSQL or MongoDB.

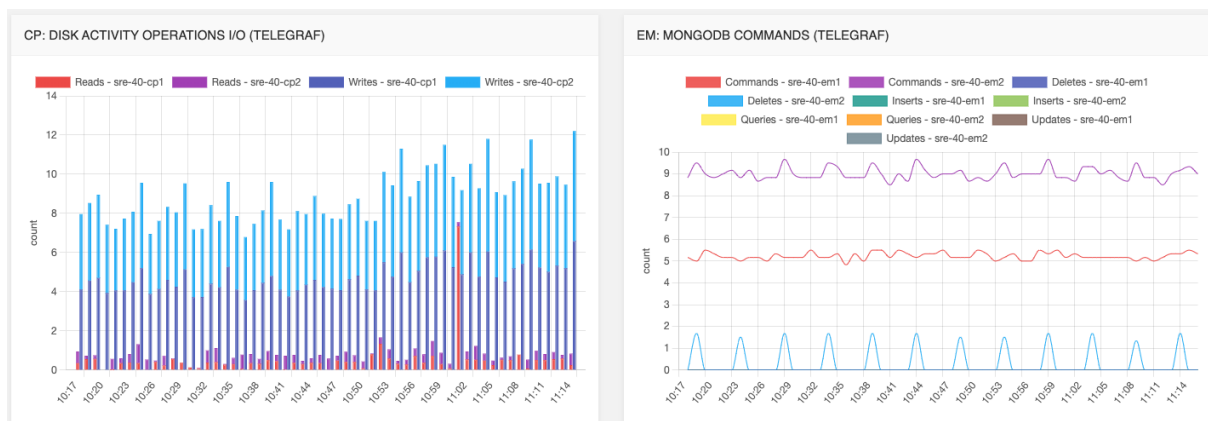


Figure 15: New Telegraf metrics

2.6 Alarms

New alarms have been introduced to monitor the number of open files, triggering when this count reaches a configured threshold. Additionally, another alarm has been added to signal when the number of open database (DB) connections reaches a configured percentage of the available DB connections.

A scheduled job has been implemented to automatically purge alarms after reaching a configured retention threshold.

Alongside the new automatic DB switchover feature introduced in this release, new alarms have been added to notify of state transitions of the *repmgrd* daemon and in the case of DB switchover.

2.7 Operations & Maintenance

2.7.1 Database Automatic Switchover

Automatic switchover of the PostgreSQL DB has been implemented on the Element Managers through the *repmgrd* daemon. In the event of a DB switchover, the Call Processors' local DBs will also be directed to follow the newly elected master. ### Events Log

A new log, located by default at `/var/log/sre/events.log`, records all state-changing events. This log offers a comprehensive view of the system and maintains a history of significant events, including alarms or changes in database state, among others. Every midnight, the current state for various items is logged to this file.

Example:

```

1 2023-12-22 00:00:00,022 CURRENT: [sre-40-em2] status.db.replication.state:
    ↪ master
2 2023-12-22 00:00:00,023 CURRENT: [sre-40-em2] status.influxDB.hosts."sre-40-em2
    ↪ ".state: pass
3 2023-12-22 00:00:00,023 CURRENT: [sre-40-em2] status.mongoDB.localMongo: True
4 2023-12-22 00:00:00,023 CURRENT: [sre-40-em2] status.mongoDB.members."
    ↪ 10.0.161.192:27017".lastHeartbeatMessage:
5 2023-12-22 00:00:00,023 CURRENT: [sre-40-em2] status.mongoDB.members."
    ↪ 10.0.161.192:27017".state: SECONDARY
6 2023-12-22 00:00:00,023 CURRENT: [sre-40-em2] status.mongoDB.members."
    ↪ 10.0.161.192:27017".syncSourceHost: 10.0.161.193:27017
7 ...
8 2023-12-22 09:34:25,356 NEW:      [sre-40-cp1] alarm.sre.process.telegraf.
    ↪ stopped: active
9 2023-12-22 09:34:30,425 CHANGE: [sre-40-cp1] alarm.sre.process.telegraf.
    ↪ stopped: active -> system-cleared
10 ...
11 2023-12-22 09:34:50,847 CHANGE: [sre-40-cp1] alarm.sre.cpu.critical: system-
    ↪ cleared -> active
12 2023-12-22 09:34:55,886 CHANGE: [sre-40-cp1] alarm.sre.cpu.critical: active ->
    ↪ system-cleared
  
```

2.7.2 Improved Summary Log

The call summary log, part of `sre.log` has been enhanced to include executions for the ENUM and HTTP interfaces, in addition to the SIP interface. For all interfaces, the switch of service logics is now explicitly detailed, and the total processing duration is also included. This information can be used to identify slow executions.

Example log for SIP, HTTP & ENUM requests:

```

1 [sre.cp.tracing.summary INFO]-2023-12-27 09:06:54,228 <sre-40-cp2> hub51u3z:
    ↪ input: 123 -> 456 (c636e7a9-5a13-4729-8d99-27fd95bffe35), logic: (SL=loop
    ↪ )Start->a->410, output: {'nit': 'sipResponse', 'responseCode': '410', '
    ↪ reasonPhrase': 'Gone', 'reasonHeader': None, 'statCode': '410'}, duration
    ↪ : 72.7 msec
2 [sre.cp.tracing.summary INFO]-2023-12-27 09:12:03,448 <sre-40-cp2> ygy1lynp:
    ↪ input: GET / , logic: (SL=RBEN HTTP )Start->crl->response->200, output: {
  
```



```

↪ 'nit': 'http', 'body': '{"message": "success"}', 'contentType': '
↪ application/json', 'headers': {}, 'responseCode': '200', 'actions': [],
↪ duration: 37.2 msecs
3 [sre.cp.tracing.summary INFO]-2023-12-27 09:15:34,344 <sre-40-cp2> k7wv2p5p:
↪ input: 32495212354 (4.5.3.2.1.2.5.9.4.2.3.e164.arpa), logic: (SL=RBEN
↪ ENUM)Start->sample enum, output: {'nit': 'enum', 'answers': [{'ttl': 900,
↪ 'order': 100, 'preference': 10, 'flags': 'u', 'service': 'sip+E2U', '
↪ regex': '!.*!domain.com!'}]}, 'actions': [], duration: 1670.9 msecs

```

2.7.3 Maintenance Jobs

By leveraging the new scheduler mechanism, maintenance tasks such as backups or file cleanup, among others, have been incorporated into the new scheduler. As a result, it is no longer necessary to configure Cron for these maintenance tasks. Parameters for these jobs have been included in the System -> Settings menu.

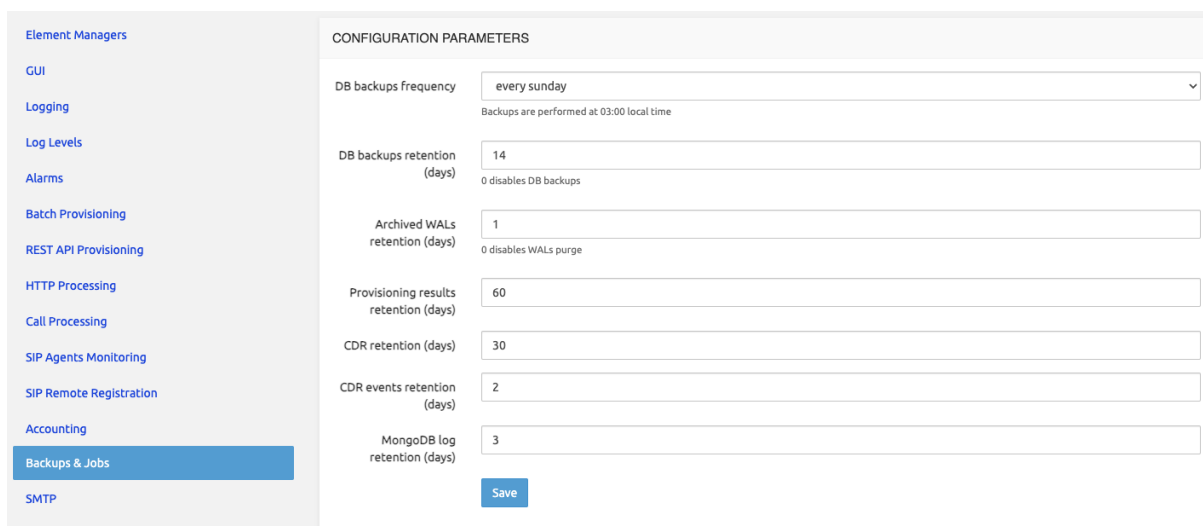


Figure 16: Jobs settings

2.7.4 Automatic Rolling Upgrade

Note

Ansible is required. Installation details are available in the **Installation Guide**.

The new SRE RPM must be acquired and downloaded on the primary EM, then run the following command on master EM:

```
1 # ansible-playbook -i /etc/ansible/sre-inventory.py /opt/sre/playbooks/upgrade.  
  ↪ yml -e src_rpm=sre-x.y.z.x86_64.rpm
```

2.8 REST API

2.8.1 Single Record Retrieval

The GET API has been enhanced to enable the return of the first matched record as a dictionary, rather than as a list of matching records when performing a query on several records.

2.8.2 Licenses Endpoint

A new API endpoint, accessible at `/licenses`, has been added to retrieve the current license usage.

Example output:

```
1 [  
2   {  
3     "name": "enum-execution-processor",  
4     "expire": "2100-01-02",  
5     "callsPerSeconds": 1,  
6     "usagePercent": 0  
7   },  
8   {  
9     "name": "http-execution-processor",  
10    "expire": "2100-01-02",  
11    "callsPerSeconds": 1,  
12    "usagePercent": 0  
13  },  
14  {  
15    "name": "sip-relay-call-processor",  
16    "expire": "2100-01-02",  
17    "callsPerSeconds": 1,  
18    "usagePercent": 0  
19  },  
20  {  
21    "name": "sip-redirect-call-processor",  
22    "expire": "2100-01-02",  
23    "callsPerSeconds": 1,  
24    "usagePercent": 0  
25  },  
26  {  
27    "name": "sip-session-execution-platform",
```

```

28     "expire": "2100-01-02",
29     "limit": 3,
30     "usagePercent": 0
31   }
32 ]
  
```

2.9 Security & Auditing

2.9.1 Password Salting

Starting from this release, local user passwords are now salted in the database. Password salting is a security measure that involves adding a unique, random value (the “salt”) to each user’s password before hashing and storing it. This prevents attackers from using precomputed tables (like rainbow tables) and enhances security by ensuring that even identical passwords result in different hashed values.

2.9.2 Service Logic Fine-Grained Access Control

By leveraging the new tagging system for Service Logics (SLs), it is now possible to define roles by specifying which SL tag a role can access. This allows for fine-grained access control of the SLs.

SERVICE LOGIC ACCESS	Read	Edit	None
Tag: release_2.0	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Tag: production	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Tag: mobile	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>

Figure 17: Service logic tags access control

2.10 GUI

2.10.1 Basic/Advanced Settings Split

In the *System -> Settings* menu, configuration parameters have been categorized into basic and advanced settings. Typically, only basic settings may need to be modified under normal conditions. Advanced settings provide more granular control over deeper components.

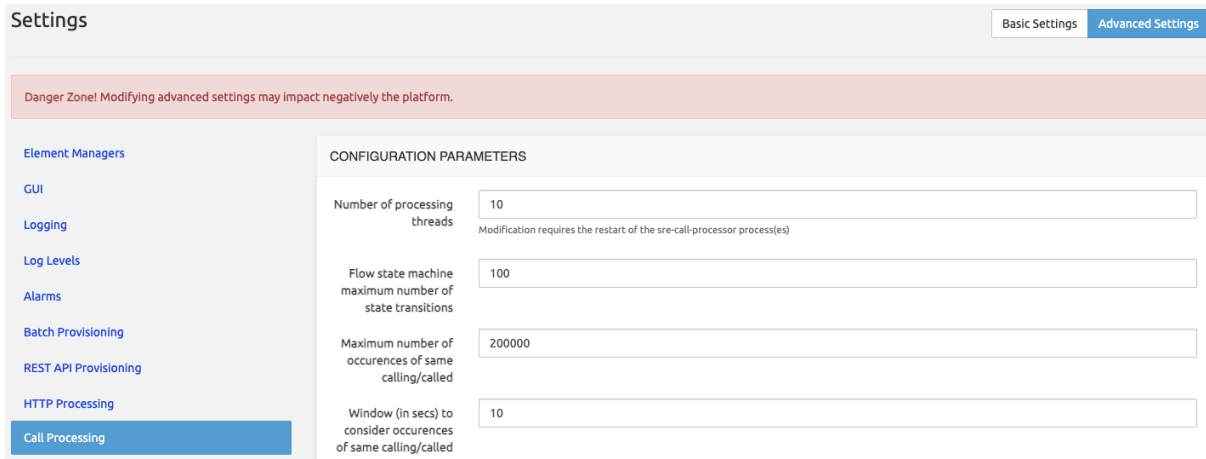


Figure 18: Basic/advanced settings

2.1.1 Miscellaneous Enhancements

The following is a list of minor enhancements which do not affect the main functionality of SRE:

- adapted REST audit log to ease parsing
- updated SREaaS Ansible syntax
- fixed top navbar
- added sre-admin tool to rename datamodel table in datamodel definition and/or service logic nodes
- added configuration parameter to modify listen/connection address of broker
- added sre-admin command to redirect traffic to sibling broker
- added sre-admin broker benchmarking tool
- added operators is NULL and is not NULL for data admin search page
- changed dashboard graphs refresh to active tab only
- added log back mechanism

3 Patch Versions Release Notes

3.1 Release 4.0.1

Pull id	Fix
1585	made SLE nodes copy/paste functionality standard

Pull id	Fix
1582	fixed API error 500 if module license is added
1575	fixed global stats to show aggregated data for all servers
1571	fixed race condition for CAC record removal when BYE arrives right after INVITE 200 OK
1569	fixed HTTP custom endpoint URL resolver to allow spaces in URL requests
1565	removed call-id-based seeding for nodes “Shuffle list of records” and “Random pick a row” to improve result randomness
1560	added missing level filter and class for alarm level warning in alarms browser
1556	forced microseconds resolution for CDR fields
1553	fixed CSV export with query
1549	fixed nodes “Create database record” and “Update database record” when trying to set an integer column to NULL
1545	fixed HTTP trace summary display
1540	fixed license loading for sre-http-processor; fixed custom endpoint resolver for sre-http-processor; pre-filled record editing with default DM values for input fields
1538	improved REST API to consider itself as master only if it is replicating to other servers; fixed validator alphanumeric to allow numbers at start
1536	fixed location of import of kamailio-local.cfg into main kamailio.cfg
1531	improved batch provisioning foreign key resolver when using numeric references
1527	adapted repmgr check to ignore error code; removed links when cloning node

3.2 Release 4.0.2

Pull id	Fix
1676	fixed regression for “Regular expression substitution” node
1673	fixed display of local databases metrics on dashboard for SREaaS
1669	fixed service logic changelog when some specific nodes are present
1654	fixed performance regression on Datamodel management page

Pull id	Fix
1650	adapted node “DB Query” to prevent empty offset value to match empty string key in call descriptor
1637	added option to control Kamailio configuration update for SREaaS deployment scripts
1635	adapted custom SIP agents probing to use the probing R-URI as key for checking live status
1633	updated registry URL for SREaaS
1629	reduced frequency of InfluxDB health checks by sre-health-monitor to reduce load
1625	fixed HTML escaping of DME constraints
1623	improved performance of node “Extract JSON path” by compiling with basic parser; removed cache expiration of compiled JSON path expressions
1610	adapted service logic tags search to apply AND logic
1608	adapted signal handling during process shutdown for process sre-cdr-collector
1589	added index on alarm table for performance
1680	fixed dashboard display of cluster status

4 Upgrade From 3.3

Note

If you are coming from a release prior to 3.3, refer to the release notes for that release to perform the intermediate steps.

As SRE 4.0 runs on a different OS version (i.e., RHEL 8) than the previous 3.x versions (i.e., RHEL 7), upgrading an existing SRE platform involves prioritizing the OS upgrade. Two possibilities exist for this upgrade:

1. Upgrade the OS in-place using RHEL conversion/upgrade tools and transition from SRE 3.3.x to SRE 4.0.x.
2. Provision new virtual machines with RHEL 8 and install the SRE RPM on top.

Option 1 minimizes operations, making it easier to maintain IP addresses and ongoing database replication, albeit with some downtime. Option 2 allows for smoother preparation but may require

more operations, especially if IP addresses need changing. In such cases, replication operations and configuration adjustments may be necessary, both within SRE and external equipment (IT systems and SIP clients).

4.1 High-Level Procedures

All the options detailed below aim to avoid any call processing downtime but differ in the duration of provisioning/monitoring downtime and the configuration changes required for external systems.

4.1.1 Option 1A

Option 1A can be performed when some provisioning/monitoring downtime is acceptable during the upgrade of the master EM. These steps must be carried out:

1. Stop SRE on standby EM
2. Upgrade the OS on standby EM
3. Upgrade the SRE RPM on standby EM
4. Stop SRE on master EM
5. Upgrade the OS on master EM
6. Upgrade the SRE RPM on master EM
7. Update the database schema on master EM
8. Start SRE on master EM
9. Start SRE on standby EM
10. Upgrade the CP one-by-one, preferably isolating them from network requests.

4.1.2 Option 1B

Option 1B should be performed when provisioning/monitoring downtime must be minimized. These steps must be carried out:

1. Stop SRE on standby EM
2. Upgrade the OS on standby EM
3. Upgrade the SRE RPM on standby EM
4. Perform a switchover of the database to promote the standby EM as master
5. Update the database schema on the new master EM
6. Start SRE on the new master EM
7. Stop SRE on the new standby EM
8. Upgrade the OS on the new standby EM

9. Upgrade the SRE RPM on the new standby EM
10. Start SRE on the new standby EM
11. Upgrade the CP one-by-one, preferably isolating them from network requests.

4.1.3 Option 2A

Option 2A can be performed when modifying configuration on other systems is possible. These steps must be carried out:

1. Provision a new master EM VM with different IP addresses and RHEL 8.
2. Install SRE on top of the OS.
3. Set up database synchronization from the SRE 3.3 master VM to this new master VM.
4. Provision a new standby EM VM with different IP addresses and RHEL 8.
5. Install SRE on top of the OS.
6. Set up database synchronization from the new master VM to this new standby VM (resulting in cascading replication: Old SRE master EM -> new SRE master EM -> new SRE standby EM).
7. Update the EM IP addresses in the configuration.
8. Shut down the old master EM VM.
9. Shut down the old standby EM VM.
10. Make the new master EM VM the primary DB.
11. Instruct all CPs to follow the new master DB.
12. Update the database schema on the new master EM.
13. Start SRE on the new master EM.
14. Start SRE on the new standby EM.
15. Instruct external IT systems to point to the new EM addresses.
16. Restart SRE on CPs one-by-one so that they communicate their logs and information to the new EMs.
17. Upgrade the CP one-by-one, preferably isolating them from network requests and updating the IP addresses on client equipment accordingly.

4.1.4 Option 2B

Option 2B can be performed when modifying configuration on other systems is difficult, and provisioning/monitoring downtime must be kept to a minimum. These steps must be carried out:

1. Provision a new standby VM with the same IP addresses and RHEL 8, in isolation mode to avoid conflicts with the existing VMs.
2. Install SRE on top of the OS.

3. Shut down the current standby EM VM.
4. Activate the network on the new standby VM.
5. Set up database synchronization from the SRE 3.3 master VM to this new standby VM.
6. Switchover the database from the master EM to the standby EM.
7. Instruct all CPs to follow the new master DB.
8. Update the database schema on the new master EM.
9. Start SRE on the new master EM.
10. Provision a new standby VM with the same IP addresses and RHEL 8, in isolation mode to avoid conflicts with the existing VMs.
11. Install SRE on top of the OS.
12. Shut down the standby EM.
13. Activate the network on the new standby EM.
14. Set up database synchronization from the new master VM to this new standby VM.
15. Start SRE on the new standby EM.
16. Upgrade the CP one-by-one, preferably isolating them from network requests.

4.2 Element Managers

4.2.1 SRE RPM Update

To launch the upgrade, on all EMs do:

```
1 # yum install /<path>/sre.4.0.x-y.x86_64.rpm
```

You must upgrade the internal DB schema. Therefore on the master EM node only, run:

```
1 # /opt/sre/bin/sre-admin db upgrade
```

The DB schema change will be applied to the other nodes through standard DB replication.

After you need to restart SRE on both EMs with:

```
1 # systemctl restart sre
```

In SRE GUI in *Settings->Licenses* set the new license keys you obtained from Netaxis Support.

Once these changes have been performed, restart SRE with the command

```
1 # systemctl restart sre
```

As maintenance jobs are now handled by the built-in scheduler, the SRE crontab file, placed by default under */etc/cron.d* should be removed

4.3 Call Processors

Call processors must be upgraded one by one.

If the call processor runs the SIP stack, perform the following steps:

1. Take the CP offline from the GUI (*System->Node operational status->out-of-service*). Alternatively, you can set the CP out-of-service from the SIP client equipment (e.g. SBC, ...). Check traffic has stopped on the CP by checking with tcpdump, sngrep or the dashboard statistics.

2. Shutdown Kamailio with:

```
1 # systemctl stop kamailio
```

3. Upgrade SRE from the RPM with the same command used for EM:

```
1 # yum install /<path>/sre.4.0.x-y.x86_64.rpm
2 # systemctl restart sre
```

4. Copy the file */opt/sre/etc/kamailio/kamailio.cfg* to */etc/kamailio*
5. Adapt the file */etc/kamailio/kamailio.cfg* depending on the deployment (usually only the line *listen*, which contains the listening address of your Kamailio instance)
6. Restart Kamailio with:

```
1 systemctl start kamailio
```

7. Enable traffic from the GUI (*System->Node operational status->in-service*)

If the call processor runs the ENUM interface or the HTTP interface, perform these steps:

1. If the client equipment allows putting the SRE CP out-of-service so that no requests are sent to it, proceed in this way.
2. Upgrade SRE from the RPM with the same command used for EM:

```
1 # yum install /<path>/sre.4.0.x-y.x86_64.rpm
```

After the upgrade is done at least on 1 CP node, make sure the CP is handling requests in the expected way, as in the previous release. Verify that CDRs are created on EMs (if enabled) for the requests handled by this CP.

If this is confirmed, proceed to the next CP node.

5 Downgrade From 4.0 to 3.3

You must downgrade the internal DB schema. Therefore on the master EM node run as user postgres:

```
1 # psql
```

and use the following commands:

```
1 # postgres=# \c sre
2 # sre=# ALTER TABLE service_logic DROP COLUMN tags CASCADE;
```

Install the previous rpm on all EMs and CPS with the command:

```
1 # yum downgrade /<path>/sre.3.3.x-y.x86_64.rpm
```

On CPs restore the previous Kamilio configuration file and restart kamilio with:

```
1 # systemctl restart kamilio
```

6 Patch Upgrade Path From 4.0.x

To upgrade to a target patch release, the Admin needs to check the upgrade path to know which actions to take.

Warning

It is important to highlight that an action needed at a patch level 4.0.N is also needed for direct upgrade to 4.0.N+1, 4.0.N+2, ...

Patch release	Needed actions
4.0.1	None

In addition to the listed needed actions:

On all nodes, do as root:

```
1 # yum update /<path>/sre.4.0.x.-y.x86_64.rpm
2 # systemctl restart sre
```

Verify always the possible differences of the following files with the diff command:

```
1 # diff /etc/kamailio/kamailio.cfg /opt/sre/etc/kamailio/kamailio.cfg
```

If any difference is observed, verify with Netaxis Support/R&D.