REST API Manual

SRE 3.3

Table of Contents

1 Introduction

This document describes the native REST API that can be used to provision data on an SRE deployment.

Netaxis SRE consists of 2 Element Managers (1 master and 1 standby) and a set of Call Processing nodes that are listening to SIP/ENUM/HTTP requests from the network. The relevant data to provide the service is stored in configurable tables of the SRE Data Model.

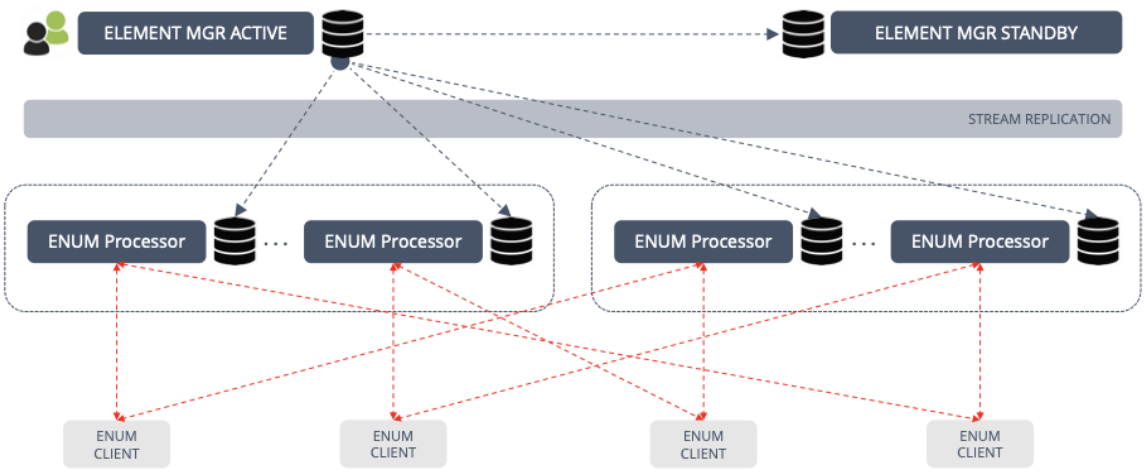

Figure 1 – Example of an SRE architecture

This document describes the generic API that can be used to provision data on the SRE deployment.

For the sake of clarity, it's worth underlining that SRE exposes two distinct interfaces:

- The **native REST API** is used for provisioning operations on individual records of the Data Model. Requests to this interface trigger direct operations on the database. This interface strictly follows the Data Model definition and is specified in 4.
- The **HTTP interface** is used for triggering a Service Logic whose actions entirely depend on the Service Logic scope and implementation. As an example, requests to this interface trigger the execution of service logics in order to ease the management of multiple records, or the addition of records in the SRE DB and on external systems at the same time. This interface is implementation dependent and therefore it's not specified in this document.

The native REST API interface can be accessed by default at port 5000, nevertheless this is configurable.

2 Data model

The Data Model is the set of tables whose structure is defined via the Data Model Editor and activated through the Datamodel Versioning tool. The tables part of the Data Model are provisioned either manually via GUI, or via Batch Provisioning (CSV), or via native REST API.

The actual Data Model in use in an SRE implementation widely depends on the requirements and details of the implementation, therefore it's not possible to generalize a model here.

It is worth noting that the native REST API interface is strictly linked to a Data Model and namely to its activated version, and it doesn't require an activation process in this context. In other words, when a Data Model version is activated on either A or B database sides (or both), the REST API definition will immediately follow the newly activated data model structure.

3 Security considerations

Both the REST and HTTP interface described below can be accessed over HTTP or HTTPS. On HTTPS, only TLS v1.2 is enabled.

Either Basic Auth or Bearer token authentication are used, which means that all requests must include either the Username/Password (Basic Auth), or the Authorization header with the following syntax:

```
1 Authorization: Bearer <token>
```

Credentials or bearer tokens are generated through the SRE GUI, with configurable access rights.

4 Supported operations

The REST API is a JSON-based API used for operations on individual records. It supports the following operations:

- **GET**: Get a single database record or a set of records
- **POST**: Create new database record(s)
- **PUT/PATCH**: Update one or more records
- **DELETE**: Delete database record(s)

5 Base URL format

The base format of request URL is:

https://<EM-address>:5000/<service>/<version>/<table>

where:

- <EM-address> is the IP address of the active Element Manager

- <service> is the service name containing the table to access (i.e., the data model name in the Data Model Editor menu)
- <version> is either active or standby, depending on the version to access. Refer to the GUI menu Settings -> Data Versioning to select the active version.
- <table> is the table to access, as defined in the SRE Data model.

The available endpoints are described below. For each endpoint, examples are provided covering different use cases.

6 Pagination

Response of a get request that return a set of records, is paginated. The total number of results *num_results*, the current page *page* and the total number of pages *total_pages* are also returned.

The following parameters can be specified in the query in order to control the returned results:

- <results_per_page> is the maximum number of results to be returned, by default 10
- <page> is the page number, by default 1

7 Filtering

Filtering is used to select only a subset of records of a get request or to limit scope of a patch/put/delete request.

The query filter is defined in a JSON object whose format is:

```
{
    "filters": [
        <condition-1>,
        <condition-2>,
        ...
    ]
}
```

where conditions are structured as:

```
{"name": <field-name>, "op": <operator>, "val": <argument>}
```

where:

- <field-name> is the name of the table field

- <operator> is one of the SQL operators ==, !=, >, <, >=, <=, in, not_in, is_null, is_not_null, like, ilike, has, any
- <argument> is the operator second argument

Additional criteria can be defined and consolidated using logical "*and*" and "*or*".

```
{"and": [{"name": <field-name>, "op": <operator>, "val": <argument>}, {"name":
    ↪ <field-name>, "op": <operator>, "val": <argument>}]}
```

The query filter can be added to the request in two ways: - by adding it to the *q* query parameter in the request url

```
Example:

`http://<base_url>/q={"filters":[{"name":"number","op":"like","val":"322%"}]}`

::: note
all special characters must be url-encoded
:::
```

- by adding it in the json body *q* key (only for non-get requests)

  Example: "' PATCH https://///

  HTTP/1.1

  { "q": {"filters": [{"name":"mycol","op":"==","val":"myvalue"}]}, "mycol":"updatedval" } "'

Warning

If filter is specified in both url and body a **400** error is returned.

8 Ordering

Response of a get request that return a set of records, can be ordered. To add ordering an *order_by* key must be set in the *q* query parameter. The *order_by* value is a list of dictionary with the following keys:

- <order-field> the name of the table field to be used to order records
- <direction> is either *asc* or *desc*

Example:

```
http://<base_url>/q={"order_by":[{"order-field":"mycolumn","direction":"asc"}]}
```

9 Endpoints

9.1 List all records

GET https://<EM-address>:5000/<service>/<version>/<table>

This request is used to list all records from a table. Response to this request uses pagination to return the records. Results can be ordered.

The following response codes can be received:

- **200** for a successful request
- **404** when the requested data was not found

9.1.1 Success

```
GET /<service>/active/<table> HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
```

200 OK

```
{
  "num_results": 1,
  "objects": [
  {
    "id": 1,
    ...
    <rest of content>
  },
  ],
  "page": 1,
  "total_pages": 1
}
```

9.2 Search records

GET https://<EM-address>:5000/<service>/<version>/<table>?q={“filter”:<filter_object>,“order_by”:<order_object>}

This request is used to list all records matching a specified query filter from a table. Results are paginated.

If no ordering is specified in the request, the results will be ordered by ascending id.

Results are paginated.

The following response codes can be received:

- **200** for a successful request
- **404** when no records matching the query filter were found

9.2.1 Success

```
GET /<service>/active/<table>?q={"filters":[{"name":"number","op":"==","val":"
    ↪ 329999999"}]} HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
```

200 OK

```
{
"num_results": 1,
"objects": [
   {
        "id": 1,
        "number": "329999999",
        ...
        <rest of content>
    }
],
"page": 1,
"total_pages": 1
}
```

9.2.2 Success order by number

```
GET /<service>/active/<table>?q={"filters":[{"name":"number","op":"like","val":
    ↪ "322816655%"}],"order_by":[{"field":"number","direction":"asc"}]} HTTP
    ↪ /1.1
Host: <EM-Host-or-FQDN>:5000
```

200 OK

```
{
  "num_results": 10,
  "objects": [
    {
      "id": 796,
      "number": "3228166550",
      <rest of content>
```

```
    },
    {
      "id": 795,
      "number": "3228166551",
      <rest of content>
    },
    {
      "id": 794,
      "number": "3228166552",
      <rest of content>
    },
    {
      "id": 793,
      <rest of content>
    },
    {
      "id": 792,
      "number": "3228166554",
      <rest of content>
    },
    {
      "id": 801,
      "number": "3228166555",
      <rest of content>
    },
    {
      "id": 800,
      "number": "3228166556",
      <rest of content>
    },
    {
      "id": 799,
      "number": "3228166557",
      <rest of content>
    },
    {
      "id": 798,
      "number": "3228166558",
      <rest of content>
    },
    {
      "id": 797,
      "number": "3228166559",
      <rest of content>
    }
  ],
```

```
  "page": 1,
  "total_pages": 1
}
```

9.2.3 No matching records found

```
GET /<service>/active/<table>?q={"filters":[{"name":"number","op":"==","val":"
    ↪ 32472801897"}]} HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
```

404 Not Found

```
{
  "code": "404",
  "message": "The path '/<service>/active/<table>' was not found."
}
```

9.3 Get a single record by id

GET https://<EM-address>:5000/<service>/<version>/<table>/<id>

This request is used to query a single record by its id.

The following response codes can be received:

- **200** for a successful request
- **404** when the requested record was not found

9.3.1 Success

```
GET /<service>/active/<table>/3446 HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
```

200 OK

```
{
  "id": 1,
  "number": "32472801896",
  <rest of content>
}
```

9.3.2 Id not found

```
GET /<service>/active/<table>/3446 HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
```

404 Not Found

```
{
  "code": "404",
  "message": "The path '/<service>/active/<table>/3446' was not found."
}
```

9.4 Create a new record

POST https://<EM-address>:5000/<service>/<version>/<table>

This request is used to create a new record. The parameters that must be specified in the body depend on the data model, in general they are the minimum set of non-nullable fields of the affected table.

The provisioning must comply with any unicity constraints on the table.

The following response codes can be received:

- **201** when the record was successfully created
- **400** when the record could not be created, for example because the unicity constraint is violated
- **500** when the input data could not be validated, for example a string was provided instead of an int

9.4.1 Success

```
POST /<service>/active/<table> HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
Content-Length: 287
{
    "number": "3227971234",
    <rest of content>
}
```

201 Created

```
{
  "id": 3446,
  "number": "3227971234",
```

```
4   <rest of content>
5 }
```

9.4.2 Failure: unique constraint violated

```
1 POST /<service>/active/<table> HTTP/1.1
2 Host: <EM-Host-or-FQDN>:5000
3 Content-Length: 287
4 {
5     "number": "3227971234",
6     <rest of content>
7 }
```

400 Bad Request

```
1 {
2   "code": "400",
3   "message": "ERROR:  duplicate key value violates unique constraint \"
      ↪ idx_2f433a21e00157b3235fab51aa470e3f4cd2e87e08dd7905ca36e734\"\nDETAIL:
      ↪ Key <key
4 details...>  already exists.\n"
5 }
```

9.4.3 Failure: data validation failed

```
1 POST /<service>/active/<table> HTTP/1.1
2 Host: <EM-Host-or-FQDN>:5000
3 Content-Length: 287
4 {
5     "number": "3227971234",
6     <rest of content>
7 }
```

500 Internal Server Error

```
1 {
2   "code": "500",
3   "message": "The server encountered an unexpected condition which prevented it
      ↪  from fulfilling the request."
4 }
```

9.5 Create multiple records

POST https://<EM-address>:5000/<service>/<version>/<table>/_bulk

This request is used to create new records in bulk. The list of records to be created must be specified in the body of the request, with each record specified as described in Create a new record.

The complete operation is performed inside a transaction so that if any record fails, the complete transaction is not applied. The error returned indicates which record failed.

In case of successful request the number of records that have been created inserted will be returned.

The following response codes can be received:

- **201** when the records were successfully created
- **400** when the records could not be created, for example because the unicity constraint is violated
- **500** when the input data could not be validated, for example a string was provided instead of an int

9.5.1 Success

```
POST /<service>/active/<table>/_bulk HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
Content-Length: 536
[
    {
        "number": "3227981234",
        <rest of content>
    },
    {
        "number": "3227987648",
        <rest of content>
    }
]
```

201 Created

```
{
  "inserted": 10
}
```

9.5.2 Failure: unique constraint violated

```
POST /<service>/active/<table>/_bulk HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
Content-Length: 536
[
    {
        "number": "3227981234",
        <rest of content>
    },
    {
        "number": "3227987648",
        <rest of content>
    }
]
```

400 Bad Request

```
{
  "code": "400",
  "message": "ERROR:  duplicate key value violates unique constraint \"
    ↪ idx_1f1978ef0f3f01867abe49bf12fe6643f38d6583a4653a2326d95e10\"\nDETAIL:
    ↪ Key <key
name and details>  already exists.\\n"
}
```

9.5.3 Failure: data validation failed

```
POST /<service>/active/<table>/_bulk HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
Content-Length: 536
[
    {
        "number": "3227981234",
        <rest of content>
    },
    {
        "number": "3227987648",
        <rest of content>
    }
]
```

500 Internal Server Error

```
{
  "code": "500",
```

```
3   "message": "The server encountered an unexpected condition which prevented it
      ↪  from fulfilling the request."
4 }
```

9.6 Edit a single record by id

PUT https://<EM-address>:5000/<service>/<version>/<table>/<id>

> **Note**
>
> **PATCH** method can also be used with the same syntax.

This request is used to update a single record specified by its id. An alternative is to update records specified by a filter criteria, see the Edit records by search criteria request.

Any parameter described in the Create a new record request can be updated. The same unicity constraints apply.

In case of successful update the full updated record will be returned.

The following response codes can be received:

- **200** when the record was successfully updated
- **400** when the record could not be updated, for example because the unicity constraint is violated
- **404** when the record was not found
- **500** when the input data could not be validated, for example a string was provided instead of an int

9.6.1 Success

```
1 PUT /<service>/active/<table>/3446 HTTP/1.1
2 Host: <EM-Host-or-FQDN>:5000
3 Content-Length: 72
4 {
5     <fields to be modified>
6 }
```

200 OK

```
1 {
2   "id": 3446,
3   <rest of content>
4 }
```

9.6.2 Failure: unique constraint violated

```
PUT /<service>/active/<table>/3446 HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
Content-Length: 33
{
    "number": "32495212366"
}
```

400 Bad Request

```
{
  "code": "400",
  "message": "ERROR:  duplicate key value violates unique constraint \"
    ↪ idx_2f433a21e00157b3235fab51aa470e3f4cd2e87e08dd7905ca36e734\"\nDETAIL:
    ↪ Key <key
details> already exists.\n"
}
```

9.6.3 Failure: data validation failed

```
PUT /<service>/active/<table>/3446 HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
Content-Length: 26
{
    <content violating the data validation>
}
```

500 Internal Server Error

```
{
  "code": "500",
  "message": "The server encountered an unexpected condition which prevented it
    ↪  from fulfilling the request."
}
```

9.6.4 Id not found

```
PUT /<service>/active/<table>/3446 HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
Content-Length: 72
{
    <content>
```

```
6 }
```

404 Not Found

```
1 {
2   "code": "404",
3   "message": "The path '/<service>/active/<table>/3446' was not found."
4 }
```

9.7 Edit records by search criteria

PATCH https://<EM-address>:5000/<service>/<version>/<table>?q={“filters”:<filter_object>}

> **Note**
>
> **PUT** method can also be used with the same syntax.

This request is used to update a set of records specified by a query filter, as described in the filtering section.

Any parameter described in the Create a new record request can be updated. The same unicity constraints apply.

In case of successful update the number of records that have been updated num_modified will be returned.

The following response codes can be received:

- **200** when at least one record was successfully updated
- **400** when the record(s) could not be updated, for example because the unicity constraint is violated
- **404** when no record matching the query filter was found
- **500** when the input data could not be validated, for example a string was provided instead of an int

9.7.1 Success

```
1 PATCH /<service>/active/<table>?q={"filters":[{"name":"number","op":"==","val":
      ↪ "3227971234"},<more filters...>]} HTTP/1.1
2 Host: <EM-Host-or-FQDN>:5000
3 Content-Length: 72
4 {
```

```
5     <request content>
6 }
```

200 OK

```
1 {
2   "num_modified": 1
3 }
```

9.7.2 No matching records found

```
1 PATCH /<service>/active/<table>?q={"filters":[{"name":"number","op":"==","val":
      ↪ "3227971235"},<more filters...>]} HTTP/1.1
2 Host: <EM-Host-or-FQDN>:5000
3 Content-Length: 72
4 {
5     <request content>
6 }
```

404 Not Found

```
1 {
2   "code": "404",
3   "message": "The path '/<service>/active/<table>' was not found."
4 }
```

9.8 Delete a single record by id

DELETE https://<EM-address>:5000/<service>/<version>/<table>/<id>

This request is used to delete a single record specified by its id. An alternative is to delete records specified by a filter criteria, see the Delete records by search criteria request.

The following response codes can be received:

- **204** when the record was successfully deleted, in that case the response has no body
- **404** when the record was not found

9.8.1 Success

```
1 DELETE /<service>/active/<table>/3446 HTTP/1.1
2 Host: <EM-Host-or-FQDN>:5000
```

204 No Content

9.8.2 Id not found

```
DELETE /<service>/active/<table>/3446 HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
```

404 Not Found

```
{
  "code": "404",
  "message": "The path '/<service>/active/<table>/3446' was not found."
}
```

9.9 Delete records by search criteria

DELETE https://<EM-address>:5000/<service>/<version>/<table>?q={"filters": [<condition-1>,<condition-2>,...]}

This request is used to delete a set of records specified by a query filter, as described in the filtering section.

In case of successful delete the number of records that have been deleted deleted_records will be returned.

The following response codes can be received:

- **200** when at least one record was successfully deleted
- **404** when no record matching the query filter was found

9.9.1 Success

```
DELETE /<service>/active/<table>?q={"filters":[{"name":"number","op":"==","val"
    ↪ :"3227972345"},<more filters...>]} HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
```

200 OK

```
{
  "deleted_records": 1
}
```

9.9.2 No matching records found

```
DELETE /<service>/active/<table>?q={"filters":[{"name":"number","op":"==","val"
    ↪ :"3227972345"},<more filters...>]} HTTP/1.1
Host: <EM-Host-or-FQDN>:5000
```

404 Not Found

```
{
  "code": "404",
  "message": "no objects found"
}
```

9.10 Kill call

9.10.1 Kill call by calling number

```
POST /killcall
Host: <EM-Host-or-FQDN>:5000
Content-Length: 100
{
    "calling":"sip:+7654321"
}
```

200 OK

```
{
  "killed": 1
}
```

This request is used to close an ongoing call from the specified calling number. A regular expression can be used. The number of closed calls is returned, if no call has been found *killed* counter is 0.

9.10.2 Kill call by called number

```
POST /killcall
Host: <EM-Host-or-FQDN>:5000
Content-Length: 100
{
    "called":"sip:+7654321"
}
```

200 OK

```
{
  "killed": 1
}
```

This request is used to close an ongoing call to the specified called number. A regular expression can be used. The number of closed calls is returned, if no call has been found *killed* counter is 0.

9.10.3 Kill call by calling and called number

```
POST /killcall
Host: <EM-Host-or-FQDN>:5000
Content-Length: 100
{
    "calling":"sip:+1234567"
    "called":"sip:+7654321"
}
```

200 OK

```
{
  "killed": 1
}
```

This request is used to close an ongoing call matching both calling and called number. A regular expression can be used for both fields. The number of closed calls is returned, if no call has been found *killed* counter is 0.

9.10.4 Kill call by callid

```
POST /killcall
Host: <EM-Host-or-FQDN>:5000
Content-Length: 100
{
    "callid":"825ADB3C-4C23-4F05-AD1C-47F5C4BE4B42"
}
```

200 OK

```
{
  "killed": 1
}
```

This request is used to close an ongoing call matching a specific callid. The number of closed calls is returned, if no call has been found *killed* counter is 0.