eCall
an F24 product

REST API

Description

Filename	F24-Schweiz_Beschreibung_REST-1.11_EN.docx
Version	1.12
Last change date	27.06.2024
Document Owner	F24 Schweiz AG, Wollerau, Switzerland
Classification	public

Contents

1. Introduction

This document describes eCall's REST API. This eCall interface allows you to send SMS messages and to retrieve their status. Please refer to section ***Connections*** for more details.
You might also find the eCall help portal useful.

2. Requirements

To send messages via the REST interface, the following conditions must be met:

- You must activate the REST interface for your eCall account in the eCall portal using the menu option *Interfaces* > *Overview*.
- Your software must be able to communicate with a REST API.

3. Authorization

You may either use your eCall account name and password for authorization or create a dedicated sub-user for the REST API in the eCall portal (*Interfaces* > *Overview*).
The eCall REST service uses basic HTTP authorization. It requires the **Authorization** header to be present in all requests from the client.
The value of the **Authorization** header must be:

Basic <*base64(your_username:your_password)*>

where <*base64(your_username:your_password)*> is the Base64 encoded string of your eCall account username or self-created API credentials (see above) and the respective password, separated by a colon.

4. API versions

We try to keep the API version as stable as possible. As such new features are usually designed with backwards compatibility in mind and will not cause the API version to change.

However, it is possible that a potential breaking change cannot be avoided. This will result in the introduction of a new API version. The older version will still be supported and maintained, but new features might only be made available in the latest version.

It's possible that support for older versions of the API will be discontinued in future updates.

Version	**Date**	**Potential breaking changes**
v1 (default)	2023-02-02	n. a.
v2	2023-12-12	Timestamp returned by "GET /api/message/{messageId}" now contains the UTC timestamp indicator (previously did not contain any time zone information) `"2023-12-12T12:30:19.523"` → `"2023-12-12T11:30:19.523Z"` Timestamp returned by "POST /api/message" is now in UTC instead of the server's local time zone for consistency. `"2023-12-12T12:30:19.523+01:00"` → `"2023-12-12T11:30:19.523Z"`

4.1. Targeting a specific API version

We offer 2 options for targeting a specific version of the API:

1. Version in the URL path
2. Version in the request headers

Omitting a version will cause the request to be handled by the default version that may change with future updates.

4.1.1. Version in the URL path

The version can be set by specifying it in the url. If the version is set in the url, this will override the version set in the request headers.

- https://rest.ecall.ch/api/{version}/

Version	Endpoints for different versions
v1	https://rest.ecall.ch/api/v1/ Example: https://rest.ecall.ch/api/v1/message
v2	https://rest.ecall.ch/api/v2/ Example: https://rest.ecall.ch/api/v2/message

– Table: Endpoints for different versions

4.1.2. *Version in the request headers*

Instead of changing the endpoint the version may be set by adding the header.

X-API-Version

```
GET /api/message/messageid HTTP/1.1
Host: rest.ecall.ch
Authorization: Basic xxxxxx
X-API-Version: 2
Content-Type: application/json
Accept: application/json
```

Version	**Valid values for the header `"X-API-Version"`**
v1	“1”, “1.0”
v2	“2”, “2.0”

– Table: Valid values for the header `"X-API-Version"`

5. Request Formats

Messages can be sent using HTTP POST with parameters and content formatted as JSON in the body. Message status can be queried using HTTP GET with the message ID in the query string.

The following rules apply:

- The **Authorization** header must be present and valid, see section ***Authorization***.
- JSON names are case-insensitive (i.e., upper/lower case does not matter).
- Enums are case-sensitive (i.e., upper/lower case matters).
- All characters must be transferred using UTF8 encoding.
- All responses from eCall to the client contain a **x-f24-request-id** HTTP header. Please provide this value for all support cases, see section ***The x-f24-request-id header***.

- **Example JSON body:**

```
{
    "channel": "Sms",
    "from": "0041769999999",
    "to": "0041768888888",
    "content": {
        "type": "Text",
        "text": "Hello eCall world :)"
    }
}
```

Important: For performance reasons, account settings parameters are cached by the REST interface. Hence, changes in the settings may only take effect in the REST interface with a certain delay.

5.1. Sending text messages

This POST message operation is used to send a single text message, to one or more receivers. The content is submitted as a JSON object in the request body.

- **Endpoint**

api/message When used with POST, this function sends a message.

5.1.1. Request

- **Required Headers**

Content-Type Supported value is “application/json”.

Authorization See section ***Authorization***.

- **Required Parameters**

`"channel"` Currently, only `"Sms"` is supported.

`"from"` Sender, up to 16 numeric or up to 11 alphanumeric characters (from ASCII 32 to ASCII 126)
Important: There are providers, that do not accept all alphanumeric characters. Please test your sender value carefully before using it in production.
Note: For Free accounts, the only valid sender address is the mobile number used for registration.

`"content"` Message to send.
Currently, only content of type `"Text"` is supported.
The maximum length of the message text is determined in the account settings (***eCall portal***: *Account Settings > SMS > Maximum number of pages).* Text which is longer than allowed by the settings will be truncated.

`"to"` Complete mobile number (in international format, e.g., 0041781234567).

`"toList"` In case, a message must be sent to more than one receiver, a list of receiver address objects may be passed in the request.
Allowed types are:

- `"Number"` – complete mobile number, in the international format, for e.g., 0041781234567.

- `"Person"` – Name of the person, as entered in the address book in the eCall web portal, in the format: [Lastname] [Firstname], without the brackets
  Important: Both the above required parameters, `"to"` and `"toList"`, should not be passed together in the same request. If passed, the validation with fail, and an error is returned to the client. Additionally, when the field `"toList"` is passed, a list of corresponding message IDs is returned in the response. Please see section ***Response structure for a successful status query*** for more details.

- **Optional Parameters**

`"notification"` Lists the addresses and specifies the events, for returning the notifications. For example, confirmation of receipt, that may be received from the provider.

`"addresses"` List of address objects.
Allowed types are:

- `"Email"`
- `"Sms"`
- `"Url"` – Web address of a client-side service that accepts HTTP GET requests. (see section ***Notification message*** below)
- `"UrlPost"` – Web address of a client-side service that accepts HTTP POST requests. (see section ***Notification message*** below)

Maximum total length of all notification addresses is 200 characters, irrespective of the number of addresses.
Note: Validation will fail at the first invalid address found, and the processing of the request is aborted, with a corresponding error returned to the client.

`"forEvent"` Value indicating when a confirmation of receipt is desired. This field is optional. Default value is `"SuccessOnly"`.
Possible values:

- `"SuccessOnly"` – (Default) Only when reception of the message has been confirmed by the device.
- `"FinalOnly"` – Confirmation of reception or failure.
- `"All"` – Same as for `"FinalOnly"`, or when the job is delayed (what counts as a delay is defined by the telecom provider).
- `"ErrorOnly"` – Only when there was an error during sending.

"options" Specifies settings specifically for a message. Otherwise, the values from the user's settings are used.

"type" Indicates the available Settings.

Allowed types are:

- **"SmsContentOptions"**

"forceGsm" The value indicates whether characters outside the Gsm character string may be used. The value can be either true or false.

"pageLimit" Defines the maximum length a message can have as pages. The maximum size is 10 pages, everything above that is automatically reduced to 10.

- **Example – with a list of different types of notification addresses**

```
{
    "channel": "Sms",
    "from": "0041769999999",
    "to": "0041798888888",
    "content": {
        "type": "Text",
        "text": "Hello from eCall, your messaging partner!"
    },
    "notification": {
        "addresses": [
            {
                "type": "Email",
                "address": "email.address@example.com"
            },
            {
                "type": "Sms",
                "address": "0041765555555"
            },
            {
                "type": "Url",
                "address": "https://example.com/ecall_notifications"
            },
            {
                "type": "UrlPost",
                "address": "https://example.com/webhook_endpoint"
            }
        ],
        "forEvent": "FinalOnly"
    }
}
```

- **Example – with a list of different types of receiver addresses**

```
{
    "channel": "Sms",
    "from": "0041769999999",
    "toList": [
        {
            "type": "Number",
            "address": "0041798888888"
        },
        {
            "type": "Person",
            "address": "Test Person"
        },
        {
            "type": "Number",
            "address": "+41785555555"
        }
    ],
    "content": {
        "type": "Text",
        "text": "Hello from eCall, your messaging partner!"
    }
}
```

- **Example – with Options**

```
{
    "channel": "Sms",
    "from": "0041769999999",
    "to": "0041798888888",
    "content": {
        "type": "Text",
        "text": "Test Message",
        "options": {
            "type": "SmsContentOptions",
            "forceGsm": true,
            "pageLimit": 1
        }
    }
}
```

Note: At most 1530 characters (GSM encoding) are allowed per message. As there are only 160 characters available in an SMS, it may be necessary to split the message into several partial messages (pages). At most 10 pages are allowed. For the receiver's device to be able to re-assemble

these partial messages into the original message, the respective sequence data is included in each page. This reduces the number of characters available per page for the actual message by 7.

5.1.2. Response

The result is returned directly as an HTTP status code and additional response structure in JSON format. Please refer to section ***HTTP status codes*** for possible HTTP status codes.

5.1.2.1. Response structure for a successful send request

The response structure contains the following parameters.

- **Parameters**

`"messageId"`	Unique identifier of the message. It may be used as an identifier in the status query.
`"messageIdList"`	Optionally a list of objects with individual message Id and its corresponding address is returned, when the parameter `"toList"` is passed in the request, to indicate multiple receivers. Please see its definition under the subsection ***Required Parameters***. In this case, the parameter `"messageId"` is left empty.
`"timestamp"`	Time when the response was sent.

- **Example – when a message sent a single address, via the `"to"` field.**

```
{
    "messageId": "97031dc9-3814-4178-a31c-1599aa83ba67",
    "timestamp": "2022-10-17T18:36:14.5175954Z"
}
```

- **Example – when the "toList" field is passed in the request with multiple receivers.**


```
{
    "messageIdList": [
      {
          "messageId": "6f69916e-bf75-49a7-bf4b-c40a60495922",
          "address": "0041798888888"
      },
      {
          "messageId": "9124c10e-731d-4d35-9305-f108ba4f4475",
          "address": "41769999999"
      },
      {
          "messageId": "e855a321-0cae-4c10-abe3-d897ebbef7fa",
          "address": "0041785555555"
      }
    ],
    "timestamp": "2022-10-17T18:36:14.5175954Z"
}
```

5.2. Status query for a single message

Besides the possibility to be informed of the status of a message via notifications (See parameter `"notification"` in the sections for sending text messages), it is also possible to query the status of a specific message using the following GET request**.**

- **Endpoint**

api/message When used with GET, this function queries the status of a message.

5.2.1. Request

- **Required URI Parameters**

`"messageId"` Unique identifier of a message, as received in the body of the response to the send request.

- **Example**

https://rest.ecall.ch/api/**message**/97031dc9-3814-4178-a31c-1599aa83ba67

Note: Status can be queried up to seven days after sending.

5.2.2. Response

The result is returned directly as an HTTP status code and an additional response structure. Please refer to section ***HTTP status codes*** for possible HTTP status codes.

5.2.2.1. Response structure for a successful status query

The response structure contains the following parameters.

- **Parameters**

`"channel"` Channel over which the message was sent.

`"messageId"` Unique identifier of the message.

`"status"` Send status of the message. Please see section ***Status codes - SMS*** for details.

`"reasonCode"` Result of the send request to the provider.

"reason"	Result text of the send request to the provider.
"timestamp"	Timestamp of the latest status update.
"from"	Sender, up to 16 numeric or up to 11 alphanumeric characters.
"to"	Complete mobile number of the recipient, in international format.
"pageTotal"	The number of pages the message required.

- **Example**


```
{
 "channel": "Sms",
 "messageId": "97031dc9-3814-4178-a31c-1599aa83ba67",
 "status": "Delivered",
 "reasonCode": 0,
 "reason": "ESME_ROK",
 "timestamp": "2022-10-17T18:36:16.243Z",
 "from": "0041769999999",
 "to": "0041798888888",
 "pageTotal": 2
}
```

5.3. Sending multiple messages

This POST message operation is used to send multiple text messages, to multiple receivers. The content is submitted as a JSON object in the request body.

- **Endpoint**

api/messageList When used with POST, this function sends messages.

5.3.1. Request

- **Required Headers**

Content-Type Supported value is “application/json”.

Authorization See section ***Authorization***.

- **Required Parameters**

`"messages"` List of message objects. A message object is comprised of the following elements:

`"channel"`

`"from"`

`"to"`

`"content"`

The specifications for these parameters are the same as found under the **Required Parameters** of the section ***Request for single message***.

- **Optional Parameters**

`"notification"` The Lists the addresses and specifies the events, for returning the notifications. The specification for this parameter is the same as found under the **Optional Parameters** of the section ***Request for single message***.

- **Example – Two messages within one request.**

```
{
    "messages": [
        {
            "channel": "Sms",
            "from": "0041769999999",
            "to": "0041798888888",
            "content": {
                "type": "Text",
                "text": "Hello from eCall, your messaging partner!"
            }
        },
        {
            "channel": "Sms",
            "from": "0041769999999",
            "to": "0041798888888",
            "content": {
                "type": "Text",
                "text": "This is a second message in the same request."
            }
        }
    ]
    "notification": {
        "addresses": [
            {
                "type": "Email",
                "address": "email.address@example.com"
            },
            {
                "type": "Sms",
                "address": "0041765555555"
            },
            {
                "type": "Url",
                "address": "https://example.com/ecall_notifications"
            },
            {
                "type": "UrlPost",
                "address": "https://example.com/webhook_endpoint"
            }
        ],
        "forEvent": "FinalOnly"
    }
}
```

5.3.2. Response

The result is returned directly as an HTTP status code and additional response structure in JSON format. Please refer to section ***HTTP status codes*** for possible HTTP status codes.

5.3.2.1. Response structure for a successful send request

The response structure contains the following parameters.

- **Parameters**

`"results"` List of result objects, one for each message in the list of `"messages"` that was sent in the send request. A result object contains an individual message Id and its corresponding address, along with any errors, that were encountered during the validation process.

A result object contains the following elements:

`"messageId"` Unique identifier of the message. It may be used as an identifier in the status query. It is only set for messages that had all the valid parameters and were accepted.

`"address"` Mobile number of the recipient, in international format.

`"error"` An error object containing the details of errors, if any, that were encountered while processing an individual message. Please see ***Error response structure*** for a detailed description.

`"messageListId"` Unique identifier of the list of messages. It may be used as an identifier in the status query.

`"timestamp"` Time when the response was sent.

- **Example – when first of the two message objects in the send request is accepted, and the other is rejected due to validation errors.**


```
{
    "results": [
        {
            "messageId": "15ad3ed5-dc5b-4e11-8adf-e87900bd228d",
            "address": "0041991234567"
        },
        {
            "address": "004188123456",
            "error": {
                "errorCode": "InvalidContent",
                "errorMessage": "Invalid content.",
                "errorDetails": {
                    "errors": [
                        {
                            "parameter": "To",
                            "messages": [
                                "'004188123456' is an invalid receiver."
                            ]
                        }
                    ],
                    "type": "Parameter"
                }
            }
        }
    ],
    "messageListId": "c28baf6b-e08b-449f-8f96-6262a9c88b9d",
    "timestamp": "2024-04-05T02:51:29.0980158+02:00"
}
```

5.4. Status query for multiple messages

Besides the possibility to be informed of the status of a message via notifications[1]. It is also possible to query the status of a list of messages that were sent via the corresponding POST request.

[1] Please see `"notification"` parameter in the sections for sending text messages.

- **Endpoint**

api/messageList When used with GET, this function queries the status of a list of messages.

5.4.1. Request

- **Required URI Parameters**

`"messageListId"` Unique identifier of a list of messages, as received in the body of the response to the send request.

- **Example**

https://rest.ecall.ch/api/**messagelist**/97031dc9-3814-4178-a31c-1599aa83ba67

Note: Status can be queried up to seven days after sending.

5.4.2. Response

The result is returned directly as an HTTP status code and an additional response structure. Please refer to section ***HTTP status codes*** for possible HTTP status codes.

5.4.2.1. Response structure for a successful status query

The response structure contains the following parameters.

- **Parameters**

`"states"` List of state objects, one for each of those messages, in the send request, that were accepted by the interface for processing. A state object contains an individual message Id and its corresponding data. Please see its definition under ***Required Parameters*** of section ***Request for multiple message request***

`"messageListId"` Unique identifier of the list of messages.

`"timestamp"` Time when the response was sent.

- **Example**


```
{
    "states": [
        {
            "channel": "Sms",
            "messageId": "1ca5df7f-27e0-4621-b9ea-d559a0d2750c",
            "status": "Delivered",
            "reasonCode": 0,
            "reason": "Ok",
            "timestamp": "2024-03-26T01:45:41.287Z",
            "from": "0041990006004",
            "to": "0041990000010",
            "pageTotal": 1
        },
        {
            "channel": "Sms",
            "messageId": "bca05c16-2b06-4b48-aff3-cbcaa342b988",
            "status": "Delivered",
            "reasonCode": 0,
            "reason": "Ok",
            "timestamp": "2024-03-26T01:45:42.293Z",
            "from": "0041990006004",
            "to": "0041990000020",
            "pageTotal": 1
        }
    ],
    "messageListId": "58f7ec29-fa61-412b-938e-6c05d5c88ecb",
    "timestamp": "2024-03-26T01:45:55.179647Z"
}
```

5.5. Notification message

As described in the sections, ***Sending text messages*** and ***Sending multiple text messages***, a parameter `"notification"` may be specified in the request consisting of a list of addresses, to which the notifications, received from the provider, may be forwarded.

In particular, the URL notification addresses, to which an HTTP GET, or an HTTP POST request is made, may be of either of the following two types:

- Notification address of type `"Url"`
  - an HTTP GET request is made, with data parameters passed in the URL, i.e., suffixed to the given URL, as a querystring. The parameters are encoded accordingly before transmitting.
- Notification address of type `"UrlPost"`
  - an HTTP POST request is made, with data parameters passed in the request body. The parameters are formatted as JSON object.

Please see examples on the following pages.

- **Parameters**

Parameter	Description	HTTP GET	HTTP POST
`Function`	Set to a fixed value: "Notification"	▪	
`ResultCode`	Status code of the message, as shown in section ***Notification Result Codes***	▪	▪
`ResultText`	Status description in plain text, as shown in section ***Notification Result Codes***	▪	▪
`Number`	Receiver address	▪	▪
`TimeStamp`	Receipt time for the sent message (dd.mm.yyyy hh:mm:ss). Time zone is CE[S]T.	▪	▪
`JobID`	Unique identification of the original message, assigned by the system during processing. (Returned in the response to the send request, c.f. section ***Response structure for a successful status query***). Only applicable to notifications sent via HTTP GET	▪	
`MessageId`	is synonym for JobID. Only applicable to notifications sent via HTTP POST		▪
`PageTotal`	Multi-page messages: Total number of pages the message has been split into	▪	▪

- **Example – HTTP GET**

```
Function=Notification&ResultCode=0&ResultText=Message+has+been+delivered&
TimeStamp=11%2E01%2E2023+08%3A21%3A50&Number=0041789999999&JobID=
97031dc9-3814-4178-a31c-1599aa83ba67&PageTotal=2
```

- **Example – HTTP POST**


```
{
    "channel": "Sms",
    "resultCode": "0",
    "resultText": "Message has been delivered",
    "timeStamp": "11.01.2023 08:21:50",
    "number": "041789999999",
    "messageId": "97031dc9-3814-4178-a31c-1599aa83ba67",
    "pageTotal": "2",
}
```


5.5.1. Read confirmation

The client software should respond with an HTTP status code of "200 OK" to confirm that the notification message request has been received successfully.

5.5.2. Notification Result Codes

ResultCode	ResultText	Description
0	Message has been delivered	Receipt of the forwarded message was confirmed by the recipient.
1	Message has been buffered	Receipt of the forwarded message was confirmed by the recipient.
2	Message has not been delivered	Receipt of the forwarded message could not be confirmed.
3	Error Code / Error Message	Defines the error on transfer to the corresponding central point.
4	Transmission OK	The send job was forwarded to the corresponding central point.

– Table: Notification Result Codes

6. Response Formats

REST API operations return standard HTTP codes as well as an additional JSON response structure in the body.

6.1. HTTP status codes

Status codes 4xx generally indicate a client error and 5xx indicate a server error.
The following codes may be returned:

Status code	Status text	Description
200	OK	Request received successfully and may be processed by the system.
400	BadRequest	This code is returned if there was a problem with the original request, e.g., a missing parameter or invalid value.
401	Unauthorized	The credentials in the Authentication header are not valid or missing.
500	InternalServerError	Indicates that an internal error occurred during the processing. Please refer to section on ***troubleshooting*** below and get in touch with our support team.

– Table: HTTP Status Codes

6.2. Status codes – SMS

Status	Description
Pending	The message is being transmitted.
Sent	The message has been sent by the provider successfully.
Delivered	The message has been received by the recipient's end device.
Failed	Sending the message failed. Please see the `"reason"` field for more information.
DeliveryFailed	The message could not be delivered to the recipient's end device.

– Table: Status Codes – SMS

6.3. Response structure

For HTTP code "200 – OK" please refer to Response structure subsection under the sections for each of the request types, for a detailed description.

6.3.1. Error response structure

For HTTP status codes indicating an error, the JSON response structure looks as follows:

- **Parameters**

`"errorCode"` Reason why sending the message failed.
Please see section ***Error codes – SMS*** for possible values.

`"errorMessage"` A high-level description of the error.

`"errorDetails"` Optional. Contains a list of the individual errors that lead to the failure as well as the error type.

- **Example** - Passing in an invalid address as **`"to"`** parameter generates the following error:


```
{
    "errorCode": "InvalidContent",
    "errorMessage": "Invalid content.",
    "errorDetails": {
        "errors": [
            {
                "parameter": "To",
                "messages": [
                    "'004177888' is an invalid receiver."
                ]
            }
        ],
        "type": "Parameter"
    }
}
```

6.3.2. Error codes – SMS

Error code	Error message	Description
InsufficientPoints	The account does not have sufficient points to send this message.	Sending this message would exceed the account's point balance.
InvalidContent	Invalid content.	Validation of the JSON in the client request failed. This is usually due to a bad parameter value. Please refer to the `"errorDetails"` for more information.
TooManyMessagesSameReceiver	Too many requests have been sent using the same receiver.	This message has been blocked to prevent spamming the addressee: The number of messages for this addressee exceeded the configured limit for this account.
TooManyMessagesSameReceiverAndContent	Too many requests have been sent using the same	This message has been blocked to prevent spamming

Error code	Error message	Description
	receiver and content.	the addressee: The number of messages with identical content for this addressee exceeded the configured limit for this account.

– Table: Errors Codes – SMS

6.4. Troubleshooting API operations

6.4.1. The x-f24-request-id header

Every request made against the eCall REST API returns a response header named **x-f24-request-id**. This header contains an opaque value that uniquely identifies the request.

If a request is consistently failing, and you have verified that the request is properly formulated, you can use this value in support cases. In your report, include the following information:

- The value of **x-f24-request-id**, as received in the response.
- The approximate time that the request was made.
- Your eCall account information: account name and account number (if known).
- The interface user (if any) you tried to send with.
- The type of operation that the request attempted.

7. Forwarding incoming SMS messages to an URL

eCall offers forwarding of incoming messages per HTTP POST request. In the eCall portal, under the settings where SMS numbers can be rented, there is the option to have incoming SMS forwarded to an HTTP(S) address as POST request. The URL, eCall should forward incoming messages to, must be configured there as well (e.g.: https://mycompany.com/incoming/sms).

When an incoming message is forwarded using POST, the request body contains a JSON string in the following format:

```
{
    "messageId": "string",
    "channel": "Sms",
    "to": "string",
    "from": "string",
    "content": {
        "text": "string",
        "type": "Text"
    },
    "timestamp": "2024-03-13T11:44:40Z"
}
```

- **Parameters**

"channel"	Currently, only **"Sms"** is supported.
"messageId"	Unique identifier of the incoming message.
"to"	The SMS was received on this eCall reception number. The forwarding was stored for this number.
"from"	Sender number. The SMS was sent from this number.
"content"	**"type"** Always "Text". **"text"** Content of the received SMS.
"timestamp"	Time of receipt in the eCall system as UTC. The format complies with ISO 8601

- **Example**


```
{
    "messageId": "6815c501-2fe1-ee11-812a-00155dfb061c",
    "channel": "Sms",
    "to": "0041766666666",
    "from": "0041777777777",
    "content": {
        "text": "Hello eCall!",
        "type": "Text"
    },
    "timestamp": "2024-03-13T11:44:40Z"
}
```

8. Connections

8.1. Access address for the eCall REST API

The REST API access for eCall is as follows:

https://rest.ecall.ch/

The service is only available over an encrypted connection using TLS 1.2 or TLS 1.3.

8.2. IP addresses for notifications (eCall to client)

Notifications may be requested about the send status of messages as described in section ***Sending text messages***. The source IP address of these notifications will be one of these:

- 193.93.208.200
- 193.93.208.149
- 193.93.208.153

9. Table directory