eCall
an F24 product

REST API

Description

Filename	F24-Schweiz_Beschreibung_REST-1.08_EN.docx
Version	1.08
Last change date	05.03.2024
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's 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	
v2	2023-12-12	Timestamp returned by "GET /api/message/{messageId}" now contains the UTC timestamp indicator (previously didn't contain any timezone 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 servers local timezone 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	Endpoint
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

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 header “X-API-Version”
v1	“1”, “1.0”
v2	“2”, “2.0”

5. Request Formats

Messages can be sent using HTTP POST with parameters and content encoded 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 :)"
    }
}
```

Note: 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

The POST message operation is used to send text messages. Content is submitted as a JSON object in the request body.

- **Function**

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

- **Required Request Headers**

Content-Type	Supported value is “application/json”.
Authorization	See section ***Authorization***.

- **Request Body – 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 ***Reply structure for a successful status query*** for more details.

- **Request Body – Optional Parameters**

`"notification"`	Addresses for forwarding a notification (e.g., confirmation of receipt) 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 100 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.

- **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!"
    }
}
```

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.

5.1.1. *Response*

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

5.1.1.1. Reply structure for a successful send query

The reply structure is in JSON format.

- **Parameters**

`"messageId"`	Unique identifier of the message. 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 section ***Request Body - Required Parameters***. In this case, the parameter **`"messageId"`** shall be left empty.
`"timestamp"`	When the send result was received.

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

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

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

```
{
    "messageId": null,
    "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

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

- **Function**

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

- **Required URI Parameters**

`<message-id>`	Unique identifier of the message, as received in the body of the response to the send query. See section ***Reply structure for a successful status query***.

- **Example**

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

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

5.2.1. Status query result

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

5.2.1.1. Reply structure for a successful status query

The reply structure is in JSON format.

- **Parameters**

`"channel"`	Channel over which the message was sent.
`"messageId"`	Unique identifier of the message from the query.
`"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
}
```

5.2.1.2. 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

5.3. *Notification message*

As described in section ***Sending text messages***, a parameter `"notification"` may be specified in the send request with addresses to which the notifications received from the provider shall be forwarded.

This section deals with URL notification addresses, to which either an HTTP GET, or an HTTP POST request shall be made, depending on the given type.

For notification addresses 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, and are encoded accordingly for transmitting.

For notification addresses of type `"UrlPost"`, an HTTP POST request is made, with data parameters passed in the request body, and are formatted as JSON.

- **Parameters**

Parameter	Description	HTTP GET	HTTP POST
Function	Set to a fixed value of "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	Number of the addressee of the original message.	●	●
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 eCall during sending. (Received in the body of the response to the send query, c.f. section ***Reply 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",
}
```

5.3.1. Read confirmation

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

5.3.2. 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. Reply Formats

REST API operations return standard HTTP codes as well as an additional JSON reply 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	Job received successfully and passed on to the provider.
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 unknown or missing.
500	InternalServerError	General error during processing.

– Table: HTTP Status Codes

6.2. *Reply structure*

For HTTP code "200 – OK" please refer to the subsections on reply structure in section ***Reply Formats*** for a description of the additional reply structure for each request type.

6.2.1. Error reply structure

For HTTP status codes indicating an error, the JSON reply 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"
    }
}
```

6.2.2. 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 receiver and content.	This message has been blocked to prevent spamming the addressee: The number of messages with identical content for this addressee exceeded the configured limit for this account.

– Table: Errors Codes SMS

6.3. Troubleshooting API operations

6.3.1. 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. Connections

7.1. 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.

7.2. 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 IP address of these notifications will be one of:

Source IP Address:	193.93.208.200 193.93.208.149 193.93.208.153
Source port:	undefined (open; 0 to 65535)

8. Table directory