eCall
an F24 product

REST API

Description

Filename	F24-Schweiz_Beschreibung_REST-1.02_EN.docx
Version	1.02
Last change date	12.04.2023
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 6 for connection 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. Request Formats

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

The following rules apply:

- The **Authorization** header must be present and valid, see Section 3.

- 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 5.3.1.

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.

4.1. 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 3.

- **Request Body – Required Fields**

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

`"from"` Sender, up to 16 numeric or up to 11 alphanumeric characters.
Note: For Free accounts, the only valid sender address is the mobile number used for registration.

`"to"`	Complete mobile number (in international format, e.g., 0041781234567).
`"content"`	Message to send. At this time, 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. **Example:** `"content": { "type": "Text", "text": "eCall test" }`

- **Request Body – Optional Fields**

`"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"` – HTTP GET request (see Section 4.3) Maximum total length of all notification addresses is (100 – number of addresses) characters. **Example:**

```
"notification": {
    "addresses": [
        {
           "type": "Email",
           "address": "donald.duck@mcduck.com"
        },
        {
           "type": "Sms",
           "address": "0041769999999"
        },
        {
           "type": "Url",
           "address": "https://myurl.com/noti"
        }

]
```

Note: Validation will fail at the first invalid address found, and processing of the request is aborted.

"forEvent" Value indicating when a confirmation of receipt is desired. This parameter 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**

```
{
    "channel": "Sms",
    "from": "0041769999999",
    "to": "0041768888888",
    "content": {
        "type": "Text",
        "text": "Hello from eCall, Uncle Scrooge!"
    },
    "notification": {
        "addresses": [
            {
                "type": "Email",
                "address": "donald.duck@mcduck.com"
            },
            {
                "type": "Sms",
                "address": "0041765555555"
            },
            {
                "type": "Url",
                "address": "https://mcduck.com/ecall_notifications"
            }
        ],
        "forEvent": "FinalOnly"
    }
}
```

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.

4.1.1. Response

The result is returned directly as an HTTP status code and additional reply structure in JSON format. Please refer to Section 5.1 for possible HTTP status codes.

4.1.1.1. Reply structure for a successful send query

The reply structure is in JSON format.

- **Parameters**

"messageId"	Unique identifier of the message. Use as identifier in the status query.
"timestamp"	When the send result was received.

- **Example**

```
{
    "messageId": "",
    "timestamp": "2022-10-17T18:36:14.5175954+02:00"
}
```

4.2. Status query

Besides the possibility to be informed of the status of a message via notifications (parameter **"notification"** in the send request, see Section 4.1), 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 (Section 4.1.1.1).

- **Example**

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

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

4.2.1. Status query result

The result is returned directly as an HTTP status code and additional reply structure. Please refer to Section 5.1 for possible HTTP status codes.

4.2.1.1. 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 4.2.1.2 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 addressee (in international format).

- **Example**

```
{
    "channel": "Sms",
    "messageId": "73dd21e4-a6fa-4f62-9fb3-3ed928c0fc80",
    "status": "Delivered",
    "reasonCode": 0,
    "reason": "ESME_ROK",
    "timestamp": "2022-10-17T18:36:16.243",
    "from": "Donald Duck",
    "to": "0041779999999"
}
```


4.2.1.2. 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.
Read	The message has been opened on the recipient's end device.
Failed	Sending the message failed. Please see the **"reason"** field for more information.

– Table: Status Codes SMS

4.3. Notification message

As described in Section 4.1, 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.
eCall supplements the notification URL with various parameters (GET call - POST is not possible). Please be aware that parameters are part of the query string and thus URL encoded.
The following parameters are included:

- **Function**

Notification Notification message function.

- **Parameters**

ResultCode Status code of the message (as shown in the table in Section 4.3.2).

ResultText Status description in plain text (as shown in the table in Section 4.3.2).

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 4.1.1.1).

PageNumber Multi-page messages: Number of the individual page the notification is for.

PageTotal Multi-page messages: Total number of pages the message has been split into.

- **Example**

Function=Notification&ResultCode=0&ResultText=Message+has+been+delivered&TimeStamp=11%2E01%2E2023+08%3A21%3A50&Number=0041789999999&JobID= 97031dc9-3814-4178-a31c-1599aa83ba67&PageNumber=1&PageTotal=2

4.3.1. Read confirmation

You should respond with HTTP status code 200 OK to confirm that you have received the notification message successfully.

4.3.2. 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

5. Reply Formats

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

5.1. 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

5.2. Reply structure

For HTTP code “200 – OK” please refer to the subsections on reply structure in Section 4 for a description of the additional reply structure for each request type.

5.2.1. 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 5.2.2 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"
    }
}
```


5.2.2. 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.

Error code	Error message	Description
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

5.3. Troubleshooting API operations

5.3.1. 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.

6. Connections

6.1. 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.

6.2. IP addresses for notifications (eCall to client)

Notifications may be requested about the send status of messages as described in Section 4.1. 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)

7. Table directory