

REST API

Description



Filename	F24-Schweiz_Beschreibung_REST-1.16.docx
Version	1.16
Last change date	18.02.2025
Document Owner	F24 Schweiz AG, Wollerau, Switzerland
Classification	public

Contents

1. Introduction.....	4
2. Requirements	4
3. Authorization	4
4. API versions	5
4.1. Targeting a specific API version	6
4.1.1. Version in the URL path	6
4.1.2. Version in the request headers.....	7
5. Request Formats.....	8
5.1. Sending text messages.....	9
5.1.1. Request	9
5.1.2. Response.....	15
5.1.2.1. Response structure for a successful send request	15
5.2. Status query for a single message.....	17
5.2.1. Request	17
5.2.2. Response.....	17
5.2.2.1. Response structure for a successful status query.....	17
5.2.2.2. Reason codes.....	19
5.3. Sending multiple messages	20
5.3.1. Request	20
5.3.2. Response.....	22
5.3.2.1. Response structure for a successful send request	22
5.4. Status query for multiple messages.....	24
5.4.1. Request	24
5.4.2. Response.....	24
5.4.2.1. Response structure for a successful status query.....	24
5.5. Notification message.....	26
5.5.1. Read confirmation.....	28

5.5.2.	Notification Result Codes	28
6.	Response Formats	29
6.1.	HTTP status codes.....	29
6.2.	Status codes	30
6.2.1.	Channel SMS.....	30
6.2.2.	Channel Pager.....	31
6.2.3.	Channel Voice	31
6.2.4.	Channel RCS.....	33
6.3.	Response structure	34
6.3.1.	Error response structure	34
6.3.2.	Error codes	35
6.4.	Header information.....	36
6.4.1.	Client Header ID	36
6.4.2.	Server Header ID.....	36
7.	Forwarding incoming SMS messages to an URL	37
8.	Connections	39
8.1.	Access address for the eCall REST API.....	39
8.2.	IP addresses for notifications (eCall to client).....	39
9.	Table directory.....	40

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	<p>Timestamp returned by "GET /api/message/{messageId}" now contains the UTC timestamp indicator (previously did not contain any time zone information)</p> <p>"2023-12-12T12:30:19.523" → "2023-12-12T11:30:19.523Z"</p> <p>Timestamp returned by "POST /api/message" is now in UTC instead of the server's local time zone for consistency.</p> <p>"2023-12-12T12:30:19.523+01:00" → "2023-12-12T11:30:19.523Z"</p>

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 [Server Header ID](#).

- **Example JSON body:**

```
{  
  "channel": "Sms",  
  "from": "0041769999999",  
  "to": "00417688888888",  
  "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 message, to one or more receivers. The content is submitted as a JSON object in the request body.

Supported channels are:

- SMS
- RCS.

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" Messaging channel used for sending the message.

Supported values are:

- "Sms"
- "Rcs"

"from" Sender.

- **Channel SMS:** 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.
- **Channel RCS:** Your RCS sender id as assigned by the RCS provider upon registration. Please contact the eCall support team to brand your RCS sender and obtain your sender id.

- "content"** Message to send.
Currently, only content of type **"Text"** is supported.
- **Channel SMS:** 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.
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.
 - **Channel RCS:**
Maximum message length is 1000 characters.

"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, 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 will 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.

"smsType"

Sets the type, an SMS may have. If the type is not set, "Standard" will be set as default. Valid types are:

"Standard" (normal SMS)

"Flash" (SMS directly shown on display)

"PrioPlus" (combination of Standard and Flash, some devices do not support it)

- **Basic example**

```
{
  "channel": "Rcs",
  "from": "MyRcsSenderId",
  "to": "00417988888888",
  "content": {
    "type": "Text",
    "text": "Hello from eCall, your messaging partner!"
  }
}
```

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

```
{
  "channel": "Sms",
  "from": "00417699999999",
  "to": "00417988888888",
  "content": {
    "type": "Text",
    "text": "Hello from eCall, your messaging partner!"
  },
  "notification": {
    "addresses": [
      {
        "type": "Email",
        "address": "email.address@example.com"
      },
      {
        "type": "Sms",
        "address": "00417655555555"
      },
      {
        "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,
      "smsType": "Flash"
    }
  }
}
```

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.
Possible values: "Sms", "Rcs", "Voice", "Pager". Please see section [Status codes](#) for an explanation of when to expect "Voice" or "Pager",

"messageId" Unique identifier of the message.

"status" Send status of the message. Please see section [Status codes](#) for details.

"reasonCode"	Result of the send request to the provider. Please see section <i>Reason codes</i> for details.
"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.

- **SMS specific parameters**

"pageTotal"	The number of pages the SMS 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  
}
```

5.2.2.2. Reason codes

Reason code	Reason
3020	Timeout while connecting to the provider (pager).
5111	Text too long (pager).
5112	Invalid receiver number (pager).
5120	Not a pager number (pager).
5121	Receiver contains invalid characters (pager).
5155	Too much traffic (pager).
5502	Syntax error.
5504	Message from provider: Operation not allowed - Maximum messages for the address exceeded.
5506	Message from provider: Analog-to-digital converter invalid.
5509	Invalid receiver number.
5510	Too many messages for same receiver from this account.
5511	Too many messages with same content for same receiver from this account.
5513	Callback, address or parts of the content are blacklisted.
5514	Blocked by out-bound filter rules.

– Table: SMS and pager reason codes

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": "Rcs",
      "from": "0041769999999",
      "to": "MyRcsSenderId",
      "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 notifications¹. 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 in the section on status queries for single messages under *Response structure for a successful status query*.

"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": "Rcs",
      "messageId": "bca05c16-2b06-4b48-aff3-cbcaa342b988",
      "status": "Delivered",
      "reasonCode": 0,
      "reason": "Ok",
      "timestamp": "2024-03-26T01:45:42.293Z",
      "from": "MyRcsSenderId",
      "to": "0041990000020"
    }
  ],
  "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	SMS only. 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

6.2.1. Channel SMS

Status	Description	Final
Pending	The message is being transmitted.	<input type="checkbox"/>
Sent	The message has been sent by the provider successfully.	<input type="checkbox"/> / <input checked="" type="checkbox"/> Depends on whether we receive a delivery status report from the provider. In general, we don't expect a report anymore after four days.
Delivered	The message has been received by the recipient's end device.	<input checked="" type="checkbox"/>
Failed	Sending the message failed. Please see the " reason " field for more information.	<input checked="" type="checkbox"/>
DeliveryFailed	The message could not be delivered to the recipient's end device.	<input checked="" type="checkbox"/>

– Table: Status Codes – SMS

6.2.2. Channel Pager

SMS messages to pager numbers are automatically sent via the pager channel.

Status	Description	Final
Pending	The message is being transmitted.	<input type="checkbox"/>
Sent	The message has been sent by the provider successfully.	<input checked="" type="checkbox"/>
Failed	Sending the message failed. Please see the " reason " field for more information.	<input checked="" type="checkbox"/>

– Table: Status Codes – Pager

6.2.3. Channel Voice

Sending voice messages over the RestApi is currently not possible. However, there is an account option for automatic conversion of SMS to landline numbers into voice messages (Swiss and German landline numbers only). If this option is set, all SMS you try to send to a Swiss or German landline number are automatically converted to voice messages, and the SMS text is read aloud to the recipient.

Notes:

- You may toggle this option in the [eCall portal](#) here: *Account Settings > SMS*.
- Sending an SMS to a landline number with the option disabled will fail.

Since the conversion only happens after the message has been accepted by the REST API and a response has been returned to you, you will never receive a voice status code in a send result. However, when you request the status of such a converted message, eCall will return a voice channel result with a voice status code.

Status	Description	Final
Pending	The message is being transmitted.	<input type="checkbox"/>
Sent	The message has been sent by the provider successfully.	<input type="checkbox"/>
Failed	Sending the message failed. Please see the " reason " field for more information.	<input checked="" type="checkbox"/>
Played	The message has been played on the recipient's end device.	<input type="checkbox"/> / <input checked="" type="checkbox"/> Depends on whether recipient hangs up without acknowledging the call.
Played_with_receipt	The message has been played on the recipient's end device, and the recipient has confirmed reception.	<input checked="" type="checkbox"/>
Next_try_pending	No connection could be established so far but eCall will try again after 15min. Reasons may be: Nobody picked up; an answer machine picked up; the line was busy; general error.	<input type="checkbox"/>

– Table: Status Codes – Voice

6.2.4. Channel RCS

Status	Description	Final
Pending	The message is being transmitted.	<input type="checkbox"/>
Sent	The message has been sent by the provider successfully.	<input type="checkbox"/>
Delivered	The message has been received by the recipient's end device.	<input type="checkbox"/>
Displayed	The message has been displayed on the recipient's end device.	<input checked="" type="checkbox"/>
Rejected	The message has been rejected by the provider.	<input checked="" type="checkbox"/>
Expired	The message's validity period has been exceeded.	<input checked="" type="checkbox"/>

– Table: Status Codes – RCS

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](#) 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

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

6.4. Header information

6.4.1. Client Header ID

To help you matching the responses to the sent requests more comfortable, it is possible to set the following header information (max. 100 characters):

x-f24-client-request-id

If this header parameter is set, it will be included into the response header, too.

6.4.2. Server Header ID

Each request made against the eCall REST API returns a response header named **x-f24-request-id**. This header contains a unique value that 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 name and/or interface user you tried to send with.
- The type of operation that the request attempted (e.g. sending a message).

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": "00417666666666",  
  "from": "00417777777777",  
  "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

– Table: Endpoints for different versions.....	6
– Table: Valid values for the header "X-API-Version"	7
– Table: SMS and pager reason codes.....	19
– Table: Notification Result Codes	28
– Table: HTTP Status Codes	29
– Table: Status Codes – SMS.....	30
– Table: Status Codes – Pager	31
– Table: Status Codes – Voice.....	32
– Table: Status Codes – RCS	33
– Table: Errors Codes	36