



M2M Services Ltd.

JSON API

for retrieving events from M2M Cloud

V3.1

Contents

Contents	2
General guidelines.....	3
Prerequisites.....	3
Token Generation.....	4
<i>Create authorization code</i>	4
<i>Create access token</i>	4
<i>Regenerate Access Token</i>	5
Events related requests.....	7
<i>Get server events</i>	7
<i>Waiting for new events in real time</i>	9

General guidelines

The M2M REST API is intended to provide an easy integration of the M2M cloud with external platforms.

This document describes how to retrieve historical events and wait for new events in real time.

The M2M REST API is based on HTTPS POST requests using JSON formatted data.

For all API calls the following settings should be used:

Content-Type: application/json

Request Type: POST

Header: Authentication: OAuth 2.0 Token based authentication

Prerequisites

- You must have a valid end user account in the RControl Client Mobile App:
 - iOS: <https://apps.apple.com/us/app/rcontrol/id1463597829>
 - Android:
<https://play.google.com/store/apps/details?id=m2m.mobile&hl=bg&gl=US&pli=1>
- It is highly recommended that you become acquainted with the basic working of the M2M communicators and platform.
- The communicator should be properly wired to the alarm panel. Please refer to the Quick Installation Manual that comes with each communicator.
- Before starting the integration via the M2M JSON API, be sure that you first receive events within the RControl Admin Portal.
- You must retrieve access token with two API requests (see section Authorizing)

Token Generation

API:	https://app.m2mservices.com/CommonAdministrationService/api/v3/xxxx		
Content Type:	application/json	Request Type:	POST
Request Format:	For the JSON format see bellow		

Create authorization code

<https://app.m2mservices.com/CommonAdministrationService/api/v3/createauthorizationcode>

Headers: None

Request:

```
{
  "UserName": "string",
  "UserPass": "string"
}
```

Success Response:

```
{
  "Success": true,
  "ErrorCode": 0,
  "ErrorMsg": null,
  "AuthCode": " a4a3854c-cb55-4b51-89ae-09057da8a886"
}
```

Note! AuthCode is the Authorization Code. It is going to be used in the next step of the authentication process.

Sample Error Responses:

```
{ "Success": false, "ErrorCode": -1001, "ErrorMsg": "ERROR: ACCESS DENIED" }
{ "Success": false, "ErrorCode": -1004, "ErrorMsg": "ERROR: AUTHENTICATION ERROR" }
{ "Success": false, "ErrorCode": -1003, "ErrorMsg": "ERROR: INVALID PARAMETERS" }
```

Create access token

<https://app.m2mservices.com/CommonAdministrationService/api/v3/createaccesstoken>

Request:

```
{
```

```
"AuthCode": "a4a3854c-cb55-4b51-89ae-09057da8a886"
}
```

Note! The AuthCode is the code returned from the “Create authorization code” request.

Success Response:

```
{
  "Success": true,
  "ErrorMsg": null,
  "ErrorCode": 0,
  "AccessToken": "bfed9dca1b55439588ab675b7c636d5c",
  "RefreshToken": "a0e63274f24e442d980763050bd59131"
}
```

Note! The Access token is required to access the rest of the API. It must be added in the header of each request.

Example header: M2MOAuth2Token: bfed9dca1b55439588ab675b7c636d5c

Sample Error Response:

```
{
  "Success": false,
  "ErrorCode": -2005,
  "ErrorString": "ERROR: INVALID AUTHORIZATION CODE",
  "AccessToken": null,
  "RefreshToken": null
}
```

Regenerate Access Token

<https://app.m2mservices.com/CommonAdministrationService/api/v3/regenerateaccesstoken>

This function generates new Access and Refresh tokens.

Header:

M2MOAuth2Token: a0e63274f24e442d980763050bd59131

Note! The “M2MOAuth2Token” token passed through the header must be the refresh token, not the authorization token. If your session has expired, you can regenerate the access token, or can just go through the initial procedure of creating authorization code and creating new access token.

Request: {}

Success Response:

```
{
```

```
"Success": true,  
"ErrorCode": 0,  
"ErrorMsg": null,  
"AccessToken": "9cfa7fe80e7a4535b12ec571b39bc5f2",  
"RefreshToken": "325ca691a9e0420e8e038d7bca599485"  
}
```

Sample Error Response:

```
{  
  "Success": false,  
  "ErrorCode": -2002,  
  "ErrorMsg": "ERROR: INVALID REFRESH TOKEN",  
  "AccessToken": null,  
  "RefreshToken": null  
}
```

Events related requests

Get server events

<https://app.m2mservices.com/CommonAdministrationService/api/v3/getserverevents>

Gets a page of events for certain device from the M2M Cloud.

Header:

M2MAuth2Token: bfed9dca1b55439588ab675b7c636d5c

Request:

```
{
  "SerialNumber": "nullable-string",
  "IMEI": "nullable-string",
  "Count": 30,
  "BeforeDate": true,
  "DateTime": unix-timestamp-ms,
  "WaitEventsMask": 2,
  "IgnoredLevelTypeCodes": ["string"],
  "AlarmCulture": "nullable-string",
}
```

"SerialNumber" or "IMEI" – identifies the device for which to retrieve the events

"Count" – the number of events to be fetched (page size). By default, it is 30.

"BeforeDate" – specifies if data filter is enabled

"DateTime" – UNIX timestamp in ms that specifies the UTC date before which we should fetch events. Applicable only if data filter is enabled

"WaitEventsMask" – should be set to 2 in order to fetch events

"IgnoredLevelTypeCodes" – specifies the event types to be ignored. Options are: "System", "Test", "Trouble", "Alarm", "Info"

"AlarmCulture" – specifies current language settings of the phone. By default, the language of the retrieved events is determined by the language of the phone. However, specific language can be set for the M2M user to override the phone language if needed.

Note!

A typical use case is to retrieve the most recent page of events with "getserverevents" and when the user scrolls to the bottom of the page to call "getserverevents" to get the next page.

To get the first page with the most recent events, use "BeforeDate" = true and "DateTime" = UTC.Now + 1 day.

To get the next page of events, use "BeforeDate" = true and "DateTime" = [timestamp of the last event from the previous page].

Note!

Periodical tests and Connect/Disconnect events usually must not be displayed to the end user, only to installers. That's why it is strongly recommended to use

"IgnoredLevelTypeCodes": ["System", "Test"].

Sample request to retrieve the first page with the most recent events.

```
{
  "IMEI": "12345678901234",
  "Count": 30,
  "BeforeDate": true,
  "DateTime": 1670081402000,
  "WaitEventsMask": 2,
  "IgnoredLevelTypeCodes": [
    "System", "Test"
  ],
  "AlarmCulture": "en-US"
}
```

Response:

```
{
  "Success": true,
  "ErrorCode": 0,
  "ErrorString": "nullable-string",
  "ResponseEventsMask": 2,
  "AlarmMessageResponse": {
    "AlarmMessagesCount": N,
    "AlarmMessages": [
      {
        "ID": 4373746094,
        "SetTime": 1669983367000,
        "Message": "human-readable event",
        "Code": "ContactId or SIA code",
        "LevelTypeCode": 4
      },
      ...
    ],
    "LastAlarmID": 4373746094
  }
}
```

```
}  
}
```

"SetTime" – UNIX timestamp in ms which represents the date and time in UTC when the event happened

"Message" – human-readable description of the event in the requested language

"Code" – ContactId or SIA event code for use by the central stations

"LevelTypeCode" – the type of the event (ex. alarm, trouble, info, test)

"LastAlarmID" – the id of the most recent message in the returned list. Used in the subsequent requests to check if there are newer events

Waiting for new events in real time

<https://app.m2mservices.com/CommonAdministrationService/api/v3/waitserverevents>

Used to fetch the newest events, newer than the provided last alarm ID.

Header:

M2MOAuth2Token: bfed9dca1b55439588ab675b7c636d5c

Request:

```
{  
  "IMEI": "string",  
  "AlarmLastAlarmID": 166393612100,  
  "AlarmFromDate": 0,  
  "AlarmToDate": 1669935659,  
  "ResultImmediately": false,  
  "WaitEventsMask": 2,  
  "IgnoredLevelTypeCodes": ["string"],  
  "AlarmCulture": "nullable-string"  
}
```

"AlarmLastAlarmID" – the id of the most recent event that was previously fetched. Used to optimize data traffic and to avoid retrieving older events

"AlarmFromDate" – must be set to 0;

"AlarmToDate" – unix timestamp that must be set to UTC.Now + one day;

"AlarmCulture" - is used to send the current language of the app, prioritizing phone's default language if the user chose default.

"ResultImmediately" – always set as false in order to block for 10 seconds and wait for new events

"WaitEventsMask" – should be set to 2 in order to fetch events

"IgnoredLevelTypeCodes" – specifies the event types to be ignored. Options are: "System",

"Test", "Trouble", "Alarm", "Info"

"AlarmCulture" – specifies current language settings of the phone. By default, the language of the retrieved events is determined by the language of the phone. However, specific language can be set for the M2M user to override the phone language if needed.

Note!

A typical use case is to retrieve the most recent page of events with "getserverevents" and then to subsequently call "waitserverevents" to wait for new events in real time while there is interaction with the end user. It will block for around 10 seconds or until a new message has arrived.

Note!

It is VERY IMPORTANT to call waitserverevents ONLY if there is active interaction with the user.

Waitserverevents MUST NEVER be called for prolonged intervals from multiple users simultaneously as this can have negative impact on the performance of the M2M cloud. For example, it must NEVER be called from a mobile application while it is on background or not in use!

Sample request to wait for new events after the last one retrieved with getserverevents:

```
{
  "IMEI": "12345678901234",
  "AlarmLastAlarmID": 4374386702,
  "AlarmFromDate": 0,
  "AlarmToDate": 1670081402000,
  "ResultImmediately": false,
  "WaitEventsMask": 2,
  "IgnoredLevelTypeCodes": [
    "System",
    "Test"
  ],
  "AlarmCulture": "us-US"
}
```

Response:

```
{
  "Success": true,
  "ErrorCode": 0,
  "ErrorString": "nullable-string",
  "SessionID": 12345,
  "ResponseEventsMask": 2,
  "WakeupReasonMask": 0,
  "AlarmMessageResponse": AlarmMessageResponse?
}
```