



M2M Services Ltd.

JSON API

for integration with the M2M communicators

V1.9

Contents

General guidelines.....	3
Prerequisites.....	3
Dispatching events to an external system	3
Contact ID messages	3
Invoking interactive features from an external system	4
Status Request	4
Arming	4
Arming in Stay	5
Disarming	5
Bypassing Zones (for selected panels)	5
Unbypassing Zones (for selected panels)	5
Get Panel Zone Status (for selected panels)	6
Get Communicator Input Status	6
Activating a Communicator Output	6
Activating an External Output (for selected panels)	7
Configuring device attributes	7
Changing Account Number	7
Infrastructure testing	8
Inject Events	8
Sample C# code for invoking M2M JSON API.....	9
Testing the API with Advanced REST Client.....	10

General guidelines

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

It allows real-time dispatching of events from the M2M communicators to external systems. It also allows to take advantage of the interactive features of the communicators such as remote arming and disarming. It also enable administrators to make some simple configurations of the device.

The M2M JSON API is based on HTTP POST request of JSON formatted data.

Prerequisites

- You must have a valid CMS (Central Monitoring Station) account in the RControl Admin Portal <https://www.m2mservices.com/admin>.
- It is highly recommended that you are 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.

Dispatching events to an external system

The external system must implement a JSON listener that will accepts HTTP POST requests of JSON formatted data.

It is highly recommended that it implements Basic Authentication, preferably via SSL.

The url of that JSON listener would be something like this:

<https://app.externalsystem.com/m2m/events>

Contact ID messages

Contact ID message from the device is in the following JSON format:

```
{"IMEI":"867755000123456", "DateTime":"2016/02/20 13:04:00",  
"ContactID":"123418160201000D"}
```

- IMEI – unique identifier of the device
- DateTime – UTC time of the event
- ContactID – event in the standard Contact ID format

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }
```

Invoking interactive features from an external system

The M2M communicators support optional interactive features such as remote arming and disarming (depending on the model of the communicator and on the panel to which it is connected).

- Before using the API, the device should be properly wired for the optional remote arming/disarming. Please refer to the Quick Installation Manual that comes with the device.
- You should have a **valid end user account** and you should have gone through the one time security pairing procedure from the RControl mobile app.
- Before using the API, it is highly recommended to check that you are able to perform the remote arming and disarming via the RControl mobile app.

The device is identified either by IMEI or by Serial Number. You can leave one of them empty.

Note!

These features require Basic Authentication via SSL. You should provide valid **end user credentials**.

Status Request

<https://app.m2mservices.com/CommonAdministrationService/api/armed>

{"IMEI":"867755000123456","SerialNumber":"12345","Partition":1}

Response JSON:

{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }

"ResultCode":0 disarmed

"ResultCode":1 armed

Arming

<https://app.m2mservices.com/CommonAdministrationService/api/arm>

{"IMEI":"867755000123456","SerialNumber":"12345","UserPIN":"0000","Partition":1,"ArmingState":1}

Response JSON:

{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }

Arming in Stay

<https://app.m2mservices.com/CommonAdministrationService/api/arm>

```
{"IMEI":"867755000123456","SerialNumber":"12345","UserPIN":"0000","Partition":1,"ArmingState":2}
```

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }
```

Disarming

<https://app.m2mservices.com/CommonAdministrationService/api/arm>

```
{"IMEI":"867755000123456","SerialNumber":"12345","UserPIN":"0000","Partition":1,"ArmingState":0}
```

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }
```

Bypassing Zones (for selected panels)

<https://app.m2mservices.com/CommonAdministrationService/api/bypass>

```
{"IMEI":"867755000123456","SerialNumber":"12345","AlarmUserId":501,"UserPIN":"0000","Partition":1,"ZonesId":[1,2]}
```

"AlarmUserId" - user paired code for this device.

"UserPIN" - user PIN for this device.

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }
```

Unbypassing Zones (for selected panels)

<https://app.m2mservices.com/CommonAdministrationService/api/upbypass>

```
{"IMEI":"867755000123456","SerialNumber":"12345","AlarmUserId":501,"UserPIN":"0000","Partition":1,"ZonesId":[1,2]}
```

"AlarmUserId" - user paired code for this device.

"UserPIN" - user PIN for this device.

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }
```

Get Panel Zone Status (for selected panels)

<https://app.m2mservices.com/CommonAdministrationService/api/getzones>

```
{"IMEI":"867755000123456","SerialNumber":"12345","Partition":1,"LoadLabel":true}
```

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null,"ZonesData":[{"ZoneID":1,"ZoneState":1,"ZoneName":"Zone 1","Partition":1,"ZoneRSSI":null},{"ZoneID":2,"ZoneState":1,"ZoneName":"Zone 2","Partition":1,"ZoneRSSI":null}]}
```

ZonesData - array of zones: {"ZoneID":1,"ZoneState":1,"ZoneName":"Zone 1","Partition":1,"ZoneRSSI":null}

ZoneState:

- 0 - ZONESTATE_DISABLED
- 1 - ZONESTATE_CLOSED
- 2 - ZONESTATE_VIOLATED
- 3 - ZONESTATE_TAMPERED
- 4 - ZONESTATE_SUPERVISION_LOST
- 5 - ZONESTATE_BYPASSED
- 6 - ZONESTATE_IN_ALARM
- 7 - ZONESTATE_TAMPER_RESTORE
- 8 - ZONESTATE_VIOLATED_RESTORE

Get Communicator Input Status

<https://app.m2mservices.com/CommonAdministrationService/api/getpins>

```
{"IMEI":"867755000123456","SerialNumber":"12345"}
```

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null,"PinsData":[{"PinID":1,"PinState":1,"PinLabel":"IN1","Disabled":false},{"PinID":2,"PinState":1,"PinLabel":"IN2","Disabled":false}]}
```

PinsData - array of pins: {"PinID":1,"PinState":1,"PinLabel":"IN1","Disabled":false}

Activating a Communicator Output

<https://app.m2mservices.com/CommonAdministrationService/api/setgpio>

```
{"IMEI":"867755000123456","SerialNumber":"12345","UserPIN":"0000","GPIOIDX":1,"Milliseconds":1000}
```

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }
```

Activating an External Output (for selected panels)

<https://app.m2mservices.com/CommonAdministrationService/api/setextgpio>

```
{"IMEI":"867755000123456","SerialNumber":"12345","AlarmUserId":501,"UserPIN":"0000","Partition":1,"GPIOIDX":1,"ExtGPIOState":1,"Milliseconds":1000}
```

"AlarmUserId" - user paired code for this device.

"UserPIN" - user PIN for this device.

"ExtGPIOState" - 0: Turn off, 1 Turn on

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }
```

Configuring device attributes

Generally, the M2M communicators are zero-configuration – in most of the cases you don't need to program the device. If any advanced programming is needed, it is normally done via the RControl Admin Portal at <https://www.m2mservices.com/admin>.

Some of the basic attributes can be also configured via the JSON API. For example, you can override the account number with which the device will report to the monitoring station.

Note!

These features require Basic Authentication via SSL. You should provide your **administrative credentials** for the RControl Admin Portal.

You will be able to make changes via JSON API only if you have permissions to make those changes via the RControl Admin Portal.

Changing Account Number

<https://app.m2mservices.com/CommonAdministrationService/api/setaccountno>

Request JSON:

```
{"IMEI":"867755000123456","SerialNumber":"12345","AccountNo":"1234"}
```

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }
```

Infrastructure testing

The M2M JSON API provides function to test the infrastructure in real-time. Events have to be formatted as ContactID and the maximum allowed events per minute is 10.

Note!

These features require Basic Authentication via SSL. You should provide your administrative credentials for the RControl Admin Portal.

You will be able to make changes via JSON API only if you have permissions and the device is assigned to an authenticated user.

Inject Events

<https://app.m2mservices.com/CommonAdministrationService/api/injectevent>

Request JSON:

```
{"IMEI":"1111111111111111","ICCID":"11111111111111111111","SerialNumber":"111111","Event":"0000181602000002","PrimaryReceiver":true}
```

Response JSON:

```
{"Success":true,"ResultCode":0,"ErrorCode":0,"ErrorString":null }
```

ErrorCode:

- 1001 "ERROR: ACCESS DENIED"
- 1003 "ERROR: INVALID PARAMETERS"
- 1014 "ERROR: NO PERMISSION FOR DEVICE"
- 1026 "ERROR: NO PERMISSION TO INJECT EVENT"
- 1027 "ERROR: CAN NOT INJECT EVENT"
- 1028 "ERROR: MAX QUOTA OF ALARMS PER MINUTE REACHED"
- 1029 "ERROR: INVALID EVENT"

Sample C# code for invoking M2M JSON API

```
public static bool IvokeM2MJsonAPI(string m2mJsonCmdUrl, string user, string pass, string jsonRequest)
{
    try
    {
        WebClient m2mProxy = new WebClient();
        m2mProxy.Headers.Add(HttpRequestHeader.ContentType, "application/json");

        string credentials = Convert.ToBase64String(ASCIIEncoding.ASCII.GetBytes(user + ":" + pass));
        m2mProxy.Headers.Add("Authorization", "Basic " + credentials);

        byte[] requestBytes = Encoding.ASCII.GetBytes(jsonRequest);
        byte[] responseBytes = m2mProxy.UploadData(m2mJsonCmdUrl, "POST", requestBytes);

        Stream stream = new MemoryStream(responseBytes);
        DataContractJsonSerializer obj = new DataContractJsonSerializer(typeof(CommandResult));
        CommandResult result = obj.ReadObject(stream) as CommandResult;

        if (result.Success)
        {
            // Handle result.ResultCode as needed
        }

        return result.Success;
    }
    catch (Exception ex)
    {
        return false;
    }
}

public class CommandResult
{
    public bool Success { get; set; }
    public int ResultCode { get; set; }
    public int ErrorCode { get; set; }
    public string ErrorString { get; set; }
}

public static void Test()
{
    string jsonUrl =
    "https://app.m2mservices.com/CommonAdministrationService/api/setaccountno";
    string jsonRequest = "{\"SerialNumber\":\"55933\",\"AccountNo\":\"1234\"}";
    IvokeM2MJsonAPI (jsonUrl, "user", "password", jsonRequest);
}
```

Testing the API with Advanced REST Client

You can also test the API directly from Chrome browser, using the Advanced REST Client plugin for Chrome. You can download and install it in Chrome from the following link:

<https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddfnphfgcellkdfbfjelloo>

This tool supports POST requests and Basic Authentication, which are two prerequisites for invoking the M2M JSON API. It also provides convenient user interface for building the requests.

“Headers” tab of the tool, showing the Basic Authorization header

Request

Method: POST Request URL: https://app.m2mservices.com/CommonAdministrationService/api/arm

Parameters

Headers

Header name: Content-Type Header value: application/json

Header name: Authorization Header value: Basic dTMxNTIzMToWMDAw

ADD HEADER

Headers are valid

Basic authorization

User name: demouser

Password: demopassword

CANCEL ACCEPT

Headers size: 68 bytes

“Body” tab of the tool, showing a sample json object for arming the system

Request

Method: POST Request URL: https://app.m2mservices.com/CommonAdministrationService/api/arm

Parameters

Body content type: application/json Editor view: Raw input

FORMAT JSON MINIFY JSON

```
{
  "IMEI": "867755000123456",
  "SerialNumber": "123456",
  "UserPIN": "0000",
  "Partition": 1,
  "ArmingState": 1
}
```