MANUFACTURING REST DOCUMENTATION

MICHAEL MEDING* , MMEDING@OUTSMARTINC.COM

CONTENTS

ABSTRACT

This documentation covers how to use the outsmart manufacturing REST API. These RESTful endpoints track MAC addresses of units that were manufactured or issue MAC addressses to devices that are being manufactured. Throughout this documentation the term BASEPATH should be replaced with "https://ops.outsmartinc.com/manufacturing" to get the fully qualified URL.

1 MANUFACTURING

1.1 ping

```
#Curl script
curl -v -X POST -H "Accept: application/json" BASEPATH/ping
```

OUTPUT (JSON)

```
1 {"status":"OK","version":1.0,"errorCode":0,"errorMsg":null,"
    fieldErrors":null,"data":null}
```

URL PATTERN
BASE-URL/ping

EXPLANATION This is as straightforward as it gets. You call this method and it returns to you a JSON status. Anything else indicates error with the server.

1.2 getNewMac

```
#Curl script
curl -v -X POST BASEPATH/getNewMac/(DeviceType)
```

OUTPUT (JSON)

```
//For this I output I used OneUpEnergyMateRev0 for DeviceType
{"status":"OK","version":1.0,"data":{"mac":"0x11E000001"}}
```

URL PATTERN
BASE-URL/getNewMac/{DeviceType}

EXPLANATION There are several device types that may be used in the URL to alter the prefix of the MAC address needed.

Request Name	MAC Prefix
OneUpEnergyMateRev0	0x11E.....
OneUpEnergyMateRev1	0x123
OneUpEnergyMateRev2	0x125
ThreeUpEnergyMateRev0	0x120
ThreeUpEnergyMateRev1	0x122
ThreeUpEnergyMateRev2	0x124
ThreeUpEnergyMateRev3	0x126
SMRev0	0x134

1.3 getNewMacs

```
#Curl script
curl -v -X POST BASEPATH/getNewMacs/(DeviceType)/{Amount}
```

OUTPUT (JSON)

```
//For this I output I used OneUpEnergyMateRev0 for DeviceType and
     10,for Amount
{"status":"OK","version":1.0,"list":["0x11E000002","0x11E000003",
    "0x11E000004","0x11E000005","0x11E000006","0x11E000007","0x11
    E000008","0x11E000009","0x11E00000A","0x11E00000B"]}
```

URL PATTERN
BASE-URL/getNewMacs/{DeviceType}/{Amount}

EXPLANATION This call is slightly different from the prior call in that you can request multiple MAC addresses at once. Be mindful that the name is

getNewMacs NOT getNewMac.

Request Name	MAC Prefix
OneUpEnergyMateRevo	0x11E
OneUpEnergyMateRev1	0x123
OneUpEnergyMateRev2	0x125
ThreeUpEnergyMateRevo	0x120
ThreeUpEnergyMateRev1	0x122
ThreeUpEnergyMateRev2	0x124
ThreeUpEnergyMateRev3	0x126
SMRevo	0x134

1.4 saveEnergyData

```
// Java RPC request
HttpRpcReq<JSONObject> req = HttpRpcFactory.newJsonRequest(url, "
    saveEnergyData");
 req.setStatusIndicator(false); // we need this to maintain backward
     compatibility
HttpRpcRes<JSONObject> response = req.invoke(jo, true, timeout);
```

OUTPUT (JSON)

```
1 {"status":"OK","version":1.0,"errorCode":0,"errorMsg":null,"
    fieldErrors":null,"data":null}
```

URL PATTERN

BASE-URL/saveEnergyData

EXPLANATION In the above Java example we stream a potentially large JSON object to the endpoint. Unfortunately we only actually care about a few of the fields from this object but will likely add more in the future. The fields that we care about are detailed below.

Type	JSON Name
String	deviceMAC
bool	deviceCalibrationPassed
String	deviceFWVersion
String	fixtureCalibrationDate
bool	deviceSensorReversedA
bool	deviceSensorReversedB
bool	deviceSensorReversedC
int	deviceNumberofports