Admin Protocol

Table of contents

Preface 3
Note 3
1. Requirements 4
2. Features 4
3. Configuration 5
3.1. Component list request 5
3.2. Partial configuration request 7
3.3. Full configuration request 10
3.4. Setting parameters 11
4. Remote API Call Request 13
4.1. Call request format 14
4.2. Response format 17
4.3. DB request example 19
5. Support 21

Preface

The information contained in this guide is subject to changes in order to improve the reliability, design or features without prior notice. Mobile Devices reserves the right to make changes in the content without obligation to notify any person or organisation of such changes or improvements. Mobile Devices can in no event be held liable for technical or editorial errors or omissions herein, nor for incidental, special or consequential damages from the furnishing, performance or use of this installation guide.

Please contact our technical support for current updates and supplemental information concerning the use and operation of this or other Mobile Devices products.

Note

Admin protocol is based on message protocol over ASN1.
Please refer to the ASN1 protocol description for further information on messages and other.

Please refer to the SDK documentation for further information on configuration parameters and API.

1. Requirements

Admin protocol is a protocol based on messages over the ASN1 protocol.

One channel is reserved for the Admin protocol: the channel 3.

It requires that the device is connected to a server based on ASN1 protocol.

On the device the following component must be at least active:

- binaryGate
  - binaryGate.port : the port of the ASN1 server
  - binaryGate.url : the url of the ASN1 server
- messageGate
  - messageGate.active must be set to 1
- adminProtocol
  - adminProtocol.active must be set to 1

2. Features

With the admin protocol you can:

- Request configuration parameters
- Set configuration parameters
- Call Remote API of components of the device

The messages are based on JSON format.

3. Configuration

3.1. Component list request

You can request the list of all the components installed on the device with the request "modnames".
It is used to request partial configuration or set parameters.

➔[Server -> Device] Request message format:

```
{
   "request": "modnames"
}
```

➔[Device -> Server] The device sends a message with the following format:

```
[
   componentName,
   ...
   componentName
]

With
        componentName = component name as JSON string
```

Example:

```
[
        "dbg",
        "ibutton",
        "update",
        "gpsMvtDetector",
        "dnsProxy",
        "sshTunnel",
        "geoFencing",
        "temperatureSensor",
        "onewire",
        "pwrManager",
        "gps",
        "criticalCommandManager",
        "acceleroMvtDetector",
        "relayControl",
        "adminProtocol",
        "netMonitoring",
        "boot",
        "assistedGps",
```

```
        "ios",
        "fileProtocol",
        "batt",
        "accelerometers",
        "messageGate",
        "gpsEcho",
        "gpsOdometer",
        "boardsInfo",
        "jvm",
        "chronoTachyGraph",
        "history",
        "leds",
        "serialPPP",
        "ledManager",
        "commandManager",
        "config",
        "modem",
        "dataEmitter",
        "dataRecorder",
        "hygrometer",
        "versionManager",
        "binaryGate",
        "pdm",
        "smartCardManager",
        "bootReason"
]
```

3.2. Partial configuration request

You can request the configuration of one or some components with the request "somemods".
You must specify the list of components as an array.

➔[Server -> Device] Request message format:

```
{
    "request": "somemods",
    "data": [
            componentName,
            … ,
            componentName
    ]
}

With
        componentName = component name as JSON string
```

Example:

```
{
    "request": "somemods",
    "data": [
            "geoFencing",
            "binaryGate"
    ]
}
```

➔[Device -> Server] The device sends a message with the following format:

```
{
    componentName: {
        "values":{
                paramName : value,
                paramName : [
                        value,
                        ...,
                        value
                ],
                ...,
                paramName : value,
        }
    },
    ...
    componentName: {
        "values":{
                paramName : value,
                paramName : [
                        value,
                        ...,
                        value
                ],
                ...
        }
    }
}
```

With

componentName = component name as JSON string
paramName = parameter name as JSON string
value= value as JSON string

Depending of the parameter, value can be a single string or an array of string.

Example:

```
{
        "binaryGate":{
                "values":{
                        "url":"193.189.124.179",
                        "forceSerialUse":"0",
                        "shd_level":"8",
                        "port":"4243"
                }
```

```
        },
        "geoFencing":{
                "values":{
                        "periodInMs":"5000",
                        "active":"0",
                        "maxLatMapResolution":"20",
                        "areaModeSearch":"0",
                        "mapsNameNotToLoad":[
                        ],
                        "maxLongMapResolution":"20",
                        "directory":[
                                "data/geofencing",
                                "/mnt/user/mmc/geofencing"
                        ]
                }
        }
}
```

3.3. Full configuration request

You can request the full configuration of the device with the request "allmods".

➔[Server -> Device] Request message format:

```
{
   "request": "allmods"
}
```

➔[Device -> Server] The device sends a message with the following format:
It is the same format as the partial request but with all the components of the device.

3.4. Setting parameters

You can set one or some parameters using the request "setval".
You must specify the component name(s), parameter name(s) and value(s).

Note: if a value is an array, you must define the full array.

➔[Server -> Device] Request message format:

```
{
   "request": "setval",
   "data": {
          componentName:{
               paramName : value,
               paramName : [
                     value,
                     ...,
                     value
               ],
               ...
          },
          ...
          componentName:{
               paramName : value,
               paramName : [
                     value,
                     ...,
                     value
               ],
               ...
          }
   }
}

With
          componentName = component name as JSON string
          paramName =  parameter name as JSON string
          value= value as JSON string
```

Example:

```
{
	"request": "setval",
	"data":{
		"gps":{
			"default_antenna_source":"0"
		},
		"geoFencing":{
			"directory":[
				"data/geofencing",
				"/mnt/user/mmc/geofencing"
			],
			"active":"0"
		}
	}
}
```

➔[Device -> Server] The device sends a message with the following format:

There is no answer to a "setval" request.

4. Remote API Call Request

Remote API calls allow the user to access the most part of the features in the devices.

You must specify:

- the component name
- the published interface of the component
- the method
- the list of the parameters, and for each parameter, its type and its value

Note: the method will be called in the same process of the Admin Protocol component. Beware when calling blocking method.

See the SDK for further information on the API that can be called.

4.1. Call request format

You must use the request "methodcall".

Note: Some APIs require that the BuffRef given in parameter to get the answer back has enough size to contain the answer. E.g for the DB API in version 2.3.6. The BuffRef can be padded with space on any other characters.

Note: if a parameter is an array, you must give the full array.

➔[Server -> Device] Request message format:

```
{
   "request": "methodcall",
   "component": componentName,
   "interface":interfaceName,
   "method":methodName,
   "params":[
         {
               "type": typeValue,
               "value": paramValue
         },
         ...,
         {
               "type": typeValue,
               "value": paramValue
         }
   ]
}

With
         componentName = component name as JSON string
         interfaceName = published interface as JSON string
         methodName = method to call as JSON string
         typeValue = type of parameter.  as JSON string
                  Type can be :
"byte", "byte[]", "char", "char[]", "BuffRef", "int", "int[]", "IntRef","boolean", "boolean[]",
"BoolRef", "String", "String[]", "long"

         paramValue = depending of type. See table Table 1
```

Type as JSON string	Value
"byte"	JSON int
"byte[]"	JSON array of JSON int
"char"	JSON char
"char[]"	JSON array of JSON char
"BuffRef"	JSON string
"int"	JSON int
"int[]"	JSON array of JSON int
"IntRef"	JSON int
"boolean"	true or false
"boolean[]"	JSON array of (true or false)
"BoolRef"	true or false
"String"	JSON string
"String[]"	JSON array of JSON string
"long"	JSON int
"void"	null

Table 1 - JSON format for value depending of type

Example 1:

To call
com.mdi.services.commandManager.CommandManager.treat (BuffRef command, BuffRef response)
with the command "status".

```
{
	"request":"methodcall",
	"component":"com.mdi.services.commandManager",
	"interface":"CommandManager",
	"method":"treat",
	"params":[
		{
			"type":"BuffRef",
			"value":"status"
		},
		{
			"type":"BuffRef",
			"value":""
		}
	]
}
```

Example 2:

To call
com.mdi.drivers.ios.DigitalOutputs.setState (int outputNumber, boolean state)
to set the output 2 to an inactive state

```
{
	"request":"methodcall",
	"component":"com.mdi.drivers.ios",
	"interface":"DigitalOutputs",
	"method":"setState",
	"params":[
		{
			"type":"int",
			"value": 2
		},
		{
			"type":"boolean",
			"value": false
		}
	]
}
```

4.2. Response format

The device sends an answer including all parameter that are of type BuffRef, IntRef or BoolRef (same order as in the method prototype) plus the result of the call.

It is given as a JSON array with type and value for each parameter and for the result.

If the method returns nothing, type is "void" and "value" is null.

The format is the following:

```
[
    {
        "type": typeValue,
        "value": paramValue
    },
    ...,
    {
        "type": typeValue,
        "value": paramValue
    }
]
```

With

typeValue = type of parameter as JSON string. See table Table 1

paramValue = depending of type. See table Table 1.

Example 1:

Result for call of com.mdi.services.commandManager.CommandManager.treat ("status","") that returns a boolean.


```
[
	{
		"type":"BuffRef",
		"value":""
	},
	{
		"type":"BuffRef",
		"value":"status,358281005317064_0005050200002820,2.27112,48.84585,108690,8,G,P,M,b,2,w,
-1,G"
	}
	,
	{
		"type":"boolean",
		"value":true
	}
]
```


Example 2:

Result for call of com.mdi.drivers.ios.DigitalOutputs.setState (2, false) that returns a boolean.


```
[
	{
		"type":"boolean",
		"value":true
	}
]
```

4.3. DB request example

Here is an example to get a value in the DB of the device.
It requires 2 remote API calls : one to get the DB key, one to get the value.

The example is based on a 2.3.6 version where the BuffRef must be padded to get the answer back.
The requested key is MDI_BOARD_ID

Request of the DB key

```
{
	"request":"methodcall",
	"component":"com.mdi.tools.config",
	"interface":"DB",
	"method":"getKey",
	"params":[
		{
			"type":"String",
			"value":"MDI_BOARD_ID"
		}
	]
}
```

Response

```
[
	{
		"type":"int",
		"value":47
	}
]
```

Request of the value

```
{
	"request":"methodcall",
	"component":"com.mdi.tools.config",
	"interface":"DB",
	"method":"get",
	"params":[
		{
			"type":"int",
			"value":47
		},
		{
			"type":"BuffRef",
			"value":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
		}
	]
}
```

Response

```
[
	{
		"type":"BuffRef",
		"value":"0005050200000282a"
	},
	{
		"type":"int",
		"value":0
	}
]
```

5. Support

For all questions not related in this guide, please contact the support team by email at support@mobile-devices.fr