

SioSdmServer

Client-server protocol

1. **Project Description**
2. **Request structure**
3. **Request commands**
4. **Response structure**
5. **Response commands**
6. **Device Templates**

1. Project Description

The main purpose of this project is a communication between clients application and remote devices using an unified interface and local server. The local server application (SioSdmServer) is a manager between clients and remote devices. Remote device could be scale, thermal camera, scanner , etc. This device is wired connected to BlueSnap and uses UART interface for data transfer.

SioSdmServer communicates to remote BlueSnap via BLE connection.

Client application connects to SioSdmServer and can communicate to remote device using internal API sent over TCP-IP.

So, there are 3 modules: client application, SioSdmServer app, BlueSnap.

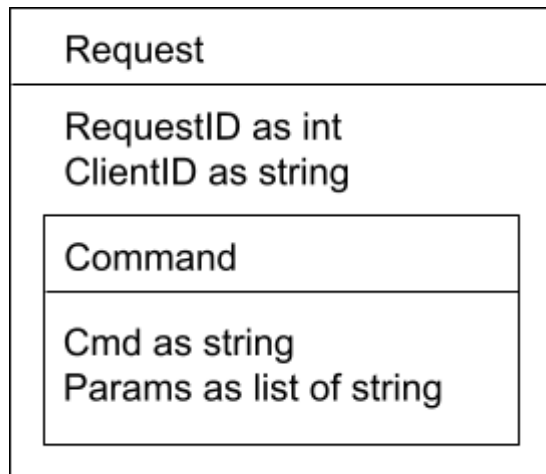


SioSdmServer application contains such modules:

- Template Download Manager: module communicates to cloud to get templates and select a template to assign for a device
- Hosted devices manager: module to communicate remote devices associated with BlueSnap by MAC address, assign a client to a remote device
- Client Manager: monitoring remote client applications
- Settings: setup TCP/IP port os server, logging management, etc

Clients communicate with the Server via basic request/response protocol TCP/IP Sockets using the JSON data-interchange format.

2. Request structure



RequestID as int

This is an identifier of request (filled in by the client), normally a number from 0 to ...

ClientID as string

This is a unique identifier string assigned to a particular application instance. Client uses this ID to identify itself on the server.

Command as object

This is an object of command which would be passed to a server.

It contains fields:

- **Cmd** as string
 - **Params** as list of string
- See list of commands for details for each command

3. Request commands

Cmd:

ClientIdentificationReq

This request sends client information to the server application. Server may do some client identification.

Server just saves ClientID information for further communication to the client and assigning the remote device to a particular client.

Params:

- *password* as string. Command specific parameter.

Example:

```
{"client_id":"ac746a5b-1c01-4a7d-80f3-db7358718ed0","command":"client_identification","parameters":{"password":"serialio"},"request_id":1}
```

Cmd:

GetRemoteDevicesReq

This is a request to the server to get a list of all server's devices. Connected and disconnected devices would be returned to the client in a response. Server sends device information and a list of support commands to the client.

Params:

- *no parameters.*

Example:

```
{"client_id":"ac746a5b-1c01-4a7d-80f3-db7358718ed0","command":"get_remote_devices","request_id":2}
```

Cmd:

DeviceConnectReq

This is a request to the server to assign a client to a particular remote device by device ID. This command doesn't allow client to establish and close physical connection to remote device. Instead of that server only associates clientID to remote deviceID, so, client will be able to get notification about remote device status and get data from the device.

Params:

- *DeviceID* as string.
- *Connect/Disconnect* as string

Example:

```
{"client_id":"ac746a5b-1c01-4a7d-80f3-db7358718ed0","command":"device_connect","parameters":{"connect_disconnect":"Connect","device_id":"66b2b20f-3e5a-4d6a-8df7-53e849d9f50a"},"request_id":3}
```

Cmd:

DeviceCommandReq

This is a request to a server with a particular command to the remote device. Response is a command status and data.

Params:

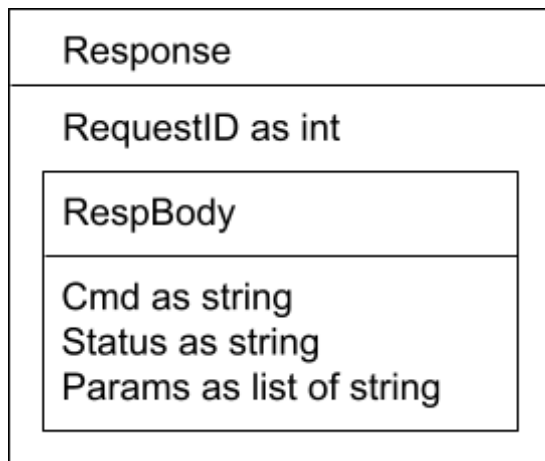
- *DeviceID* as string
- *Command* as string

Example:

```
{"client_id":"ac746a5b-1c01-4a7d-80f3-db7358718ed0","command":"device_command","parameters":{"command_name":"Weight","device_id":"66b2b20f-3e5a-4d6a-8df7-53e849d9f50a"},"request_id":4}
```

Clients send each request as a structure serialized to JSON string.

4. Response structure



RequestID as int

RespBody is a structure with fields:

- Cmd as string
- Status as string
- Params as list of string

5. Response commands

Cmd:

ClientIdentificationReq

Status as string

Example:

```
{"command": "client_identification", "request_id": 1, "status": "PASSED"}
```

Cmd:

GetRemoteDevicesReq

Status as string

Params:

- ***RemoteDevices*** structure

```
data class RemoteDevices(  
    var request_id: String, //incremental counter. Changed by Client App  
    var command: String, // "get_remote_devices"  
    var status: String = // "PASSED"  
    var remote_devices: List<RemoteDevice>  
)
```

```
data class RemoteDevice(  
    var device_id: String, // Unique device id  
    var device_type: String, // Type from Server Template
```

```

var device_name: String, // Name from Server Template
var device_state: String, // The current connection state of the remote device
var device_commands: List<DeviceCommand> // The List of saved devices on the Server
(currently connected and not connected)
)

data class DeviceCommand(
    var cmd_name: String, //This is a value which the application sends to the server. Name of
    command, not a real command for a remote device.
    var cmd_description: String //The description of command, e.g. "weight", "zero weight",...
    . Useful for clients to test remote devices by trigger button.
)

```

Example:

```

{"command": "get_remote_devices", "parameters": {"remote_devices": [{"device_commands": [{"cmd_
description": "Print the weight", "cmd_name": "Weight"}, {"cmd_description": "Zero gross
weight", "cmd_name": "Zero"}, {"cmd_description": "Save and clear
tare", "cmd_name": "Tare"}], "device_id": "66b2b20f-3e5a-4d6a-8df7-53e849d9f50a", "device_name": "
Optima OP-900B
MS", "device_state": "Disconnected", "device_type": "scale"}, {"device_commands": [{"cmd_description
": "scan
tag", "cmd_name": "scan"}], "device_id": "37764c2f-0fb1-4626-9229-2677516416ec", "device_name": "
Scanfob®
scanner", "device_state": "Disconnected", "device_type": "scanner"}]}, "request_id": 3, "status": "PASSE
D"}

```

Cmd

DeviceConnectionStatus

This command is sent by server when the status of the assigned remote device has changed.

Status as string

Params:

- DeviceID as string
- DeviceStatus as string. Possible values: "disconnected", "connected", "connecting", "connection fail", "connection lost", "disconnecting", "reconnecting", "device is unavailable", "device is ready".

Example:

```

{"command": "device_connect", "parameters": {"device_id": "66b2b20f-3e5a-4d6a-8df7-53e849d9f50a
", "device_state": "connected"}, "request_id": -1, "status": "PASSED"}

```

Cmd

DeviceCommandReq

Status as string

Params:

- DeviceID as string
- device_command as string

- *status* as string
- *command_result* as string

Example:

```
{ "command": "device_command", "parameters": { "command_name": "Weight", "command_result": "-7 lb", "device_id": "66b2b20f-3e5a-4d6a-8df7-53e849d9f50a" }, "request_id": 4, "status": "PASSED" }
```

Server receives request from client, identifies client, stores *requestID* and *clientID* and sends command to the command processing manager. The command processing manager identifies the command. If it is DeviceCommandReq then this command passes to the Template Manager. The template manager creates a device specific command using a template assigned to the remote device.

6. Device Templates

The template file is a list of rules for working with remote devices packed in JSON.

The list of templates is stored on the Server and provided by the customer.

Device template contains: device name, device type, list with command values of remote connected device, and rules for decoding command responses.

Data structures:

```
data class TemplateDto(
    val devices: List<DeviceDto>
)
```

```
data class DeviceDto(
    val device_name: String,
    val device_type: String,
    val commands: List<CommandDto>
)
```

```
data class CommandDto(
    val cmdname: String,
    val description: String,
    val data: String,
    val responses: List<ResponseDto>
)
```

```
data class ResponseDto(
    val data: String,
    val status: String
)
```

Sample of template JSON:

[

```

{
  "device_name": "Scanfob® scanner",
  "device_type": "scanner",
  "commands": [
    {
      "cmdname": "scan",
      "description": "scan tag",
      "data": "scan",
      "responses": [
        {
          "status": "PASSED",
          "data": "regex to parse data"
        },
        {
          "status": "FAILED",
          "data": "ERR"
        }
      ]
    }
  ]
},
{
  "device_name": "Ranger 3000",
  "device_type": "scale",
  "commands": [
    {
      "cmdname": "WT",
      "description": "weight",
      "data": "P\r\n",
      "responses": [
        {
          "status": "PASSED",
          "data": "regex to parse number and metric"
        },
        {
          "status": "UNSTABLE",
          "data": "regex to check data started with '(unstable)'"
        },
        {
          "status": "FAILED",
          "data": "ERR"
        }
      ]
    }
  ]
},
{
  "device_name": "Optima OP-900B MS",

```

```

"device_type":"scale",
"commands":[
  {
    "cmdname":"Weight",
    "description":"Print the weight",
    "data":"P\r\n",
    "responses":[
      {
        "status":"PASSED",
        "data":"regex to parse number and metric"
      },
      {
        "status":"UNSTABLE",
        "data":"regex to check data started with '(unstable)'"
      },
      {
        "status":"FAILED",
        "data":"ERR"
      }
    ]
  },
  {
    "cmdname":"Zero",
    "description":"Zero gross weight",
    "data":"Z\r\n",
    "responses":[
      {
        "status":"PASSED",
        "data":"OK"
      },
      {
        "status":"FAILED",
        "data":"ERR"
      }
    ]
  },
  {
    "cmdname":"Tare",
    "description":"Save and clear tare",
    "data":"T\r\n",
    "responses":[
      {
        "status":"PASSED",
        "data":"OK"
      },
      {
        "status":"FAILED",
        "data":"ERR"
      }
    ]
  }
]

```



```
    }  
  ]  
}  
]  
}
```