# SPC Web Gateway Specification

Revision 1.0

# History Record

| Revision | Date | Author | Comment |
|----------|------|--------|---------|
| 0.9 | 5-Jan-2014 | Göran Lundquist, Lundix IT | Draft |
| 1.0 | 27-Jan-2014 | Göran Lundquist, Lundix IT | First edition |

©2014 Lundix IT

Lundix IT
Renvägen 22
S-433 70 Sävedalen
Sweden
goran.lundquist@lundix.se

# Contents

# 1 Introduction

## 1.1 Purpose of the document

The purpose of this document is to describe Lundix IT SPC Web Gateway that is providing a generic web interface to Siemens SPC intrusion panels.

## 1.2 Scope

The SPC Web Gateway is based on the functionality in EDP protocol version 2.0 and SPC panel firmware 3.2. In this version of the software the scope is limited to only support TCP as transport layer and following EDP functions are **not supported**:

- • UDP as transport layer to SPC
- • EDP version 1 frame layout
- • EDP over GPRS
- • EDP Audio Commands
- • EDP Encryption
- • Door Control Commands
- • Set Clock Command
- • Set User Pin Command

## 1.3 Document References

| Id | Description | Revision |
|---|---|---|
| [EDP_SPEC] | Siemens EDP Protocol Specification | 2.0 |
| [SPC_INST_CONF] | Siemens SPC42xx/43xx/52xx/53xx/63xx, Installation & Configuration Manual | 3.2 |

## 1.4 Terminology and Abbreviations

| Term | Description |
|---|---|
| EDP | Siemens Enhanced Datagram Protocol |
| JSON | JavaScript Object Notation |
| REST | Representational State Transfer |
| RCD | Remote Communications Device |
| SIA | Security Industry Association |
| SPC panel | Siemens SPC intrusion panel |
| TCP | Transmission Control Protocol |
| URL | Uniform Resource Locator |
| WebSocket | Two-way TCP protocol RFC 6455 |
| XML | Extensible Markup Language |

## 1.5 JSON Element Data Types

All values in the JSON messages are sent as double-quoted UTF-8 strings of different "data types".
Following data types will be used in this document:

| Data Type | Description | Example |
| --- | --- | --- |
| Integer | Integer value, zero or positive, unless stated otherwise | "0", "1", "999" |
| Float | Float value | "0.1", "1.99", "-999.99" |
| String | UTF-8 String value | "User 1", "Area 1" |
| Hexadecimal | Hexadecimal string | "0123456789ABCDEF" |
| Timestamp | POSIX time defined as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970 | "1367404200" (= 1 May 2013 10:30:00 GMT) |

# 2 General Presentation of the SPC Web Gateway

## 2.1 Overview

The SPC Web Gateway is providing a generic web interface to Siemens SPC panels. The main purpose is to simplify SPC integration with third party applications and products such as Home and Building Automation Systems, Smartphone App's and Web applications. The Web API is using HTTP and REST principles (RESTful) for requests to SPC panel and WebSocket to reporting events from the SPC panel.

## 2.2 Software Architecture
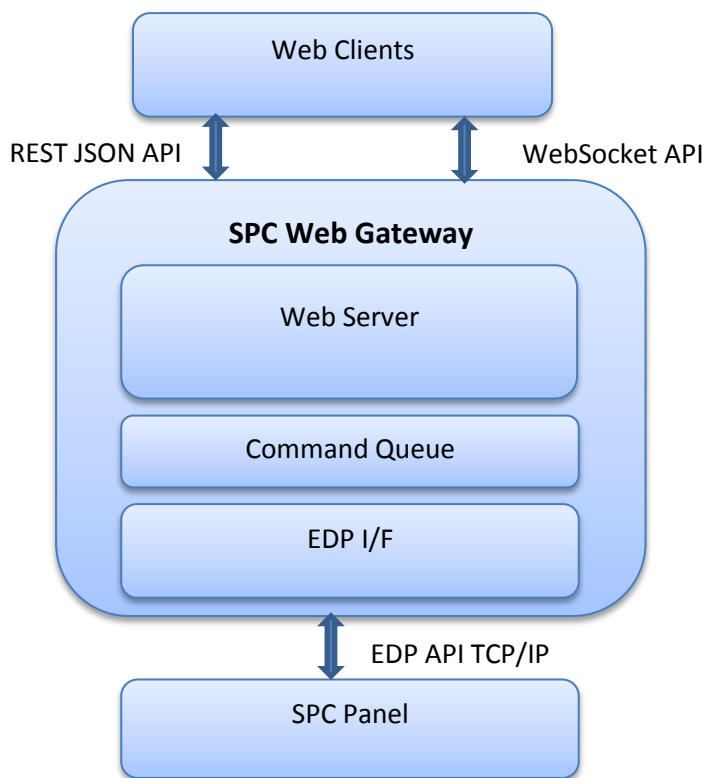


The SPC Web Gateway communicates (southbound) to the SPC panel over network using Siemens EDP protocol and offers a northbound interface - referred as Web Server - used by applications or web pages - referred as Web Clients - to operate and get information from the SPC panel. A REST JSON API is used for requests and commands and a WebSocket API for sending SIA events to the Web Clients.

The SPC Web Gateway supports multiple concurrent clients. In case the clients simultaneously send requests, the requests will be queued in the SPC Web Gateway waiting for the reply from the SPC panel. (The SPC panel is restricted to process only one request at a time). Events from the SPC panel (SIA events) are broadcasted to all "subscribing" recipients.

The SPC Web Gateway has following main features:

- Web Server. This is the core of the SPC Web Gateway.

- EDP I/F. Interface for communication to the SPC panel.

- Command Queue. Is managing the queue of pending commands and requests.

All features are embedded in one single multithreaded application with a very small footprint.

# 3  REST JSON API

The SPC Web Gateway REST JSON API provides access to the SPC panel resources via URL paths. The API uses JSON as its communication format, and the standard HTTP methods GET and PUT (see API descriptions below for which methods are available for each resource). URLs for the REST API resources have the following structure:

```
http://<spc_web_gateway_host>:<port>/spc/<resource>
```

Default <port> is 8088, but it is configurable. <resource> can be divided in several parts separated by slashes (/).

## 3.1  Response Format

All response messages consist of a status and a data part with following JSON structure:

```
{
   "status":"<return status>",
   "data":{
      "element1":"value1",
      "element2":"value2",
      …
      "elementN":"valueN"
      }
}
```

Encoding is UTF-8.

## 3.2  Response Status

Response status is returned using standard HTTP error code syntax in the header and additional info in the body of the return call, JSON-formatted.

### 3.2.1  success

If all went well the response status will be **success** and some valid data will be returned in the data part.

```
Header
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

Body
{
      "status": "success",
      "data": {
            <valid data>
      }
}
```

### 3.2.2  error

If an error occurred in processing the request the response status will be **error** and the data part will contain an error code and error message.

```
Header
```

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=utf-8

Body
{
      "status": "error",
      "data": {
            "code": "<error_code>",
            "message": "<error_message>"
      }
}
```

| Error Code | Error Message |
|---|---|
| 242 | Invalid parameters |
| 252 | Panel is in full engineer mode |
| 253 | Command is not possible now |
| 254 | Command is not permitted |
| 255 | Command is not implemented |

## 3.3   Area Resources

### 3.3.1   Area Requests

| Method | Resource | Description |
|---|---|---|
| GET | area | Returns the status of all areas |
| GET | area/<ID> | Returns the status of area <ID>. |
| GET | area/<ELEMENT> | Returns <ELEMENT> value of all areas. |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ID | The area id. (1 – number of defined areas) |
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

There will be one instance of these elements for each area.

| Element | Type | Description |
|---|---|---|
| id | Integer | Area ID (1 to maximum number of areas) |
| name | String | Area name. |
| mode | Integer | Area mode:<br>0 – area is unset<br>1 – area is part set A<br>2 – area is part set B<br>3 – area is full set |
| last_set_time | Timestamp | Last time the area was full set. Present only if available. |
| last_set_user_id | Integer | ID of user that last full set the area, 1 to 515. Present only if available. |

| last_set_user_name | String | Name of user that last full set the area. Present only if available. |
|---|---|---|
| last_unset_time | Timestamp | Last time the area was unset. Present only if available. |
| last_unset_user_id | Integer | ID of user that last unset the area, 1 to 515. Present only if available. |
| last_unset_user_name | String | Name of user that last unset the area. Present only if available. |
| last_alarm | Timestamp | Timestamp of last alarm. Present only if available. |

## Example: GET area

```
{
     "status":"success",
     "data":{
          "area": [
               {
                    "id": "1",
                    "name": "Area 1",
                    "mode": "0",
                    "last_set_time": "1390068017",
                    "last_set_user_id": "1",
                    "last_set_user_name": "User 1",
                    "last_unset_time": "1390068017",
                    "last_unset_user_id": "2",
                    "last_unset_user_name": "User 2",
                    "last_alarm":"1389030757"
               },
               {
                    "id":"2",
                    "name":"Area 2",
                    "mode":"0",
                    "last_set_time":"1390068017",
                    "last_set_user_id":"1",
                    "last_set_user_name":"User 1",
                    "last_unset_time":"1390068017",
                    "last_unset_user_id":"2",
                    "last_unset_user_name":"User 2"
               }
          ]
     }
}
```

## Example: GET area/1

```
{
     "status":"success",
     "data":{
          "area": [
               {
                    "id": "1",
                    "name": "Area 1",
                    "mode": "0",
                    "last_set_time": "1390068017",
                    "last_set_user_id": "1",
                    "last_set_user_name": "User 1",
                    "last_unset_time": "1390068017",
                    "last_unset_user_id": "2",
```

```
                            "last_unset_user_name": "User 2",
                            "last_alarm":"1389030757"
                    }
            ]
        }
}
```

**Example: GET area/name**

```
{
        "status":"success",
        "data":{
                "area":[
                        {
                                "name":"Area 1"
                        },
                        {
                                "name":"Area 2"
                        }
                ]
        }
}
```

### 3.3.2 Area Commands

| Method | Resource | Description |
|---|---|---|
| PUT | area/<ID>/set | Full sets area <ID> |
| PUT | area/<ID>/set_a | Part sets A area <ID> |
| PUT | area/<ID>/set_b | Part sets B area <ID> |
| PUT | area/<ID>/unset | Unsets area <ID> |

Resource Parameters

| Parameter | Description |
|---|---|
| ID | The area id. (1 – number of defined areas) |

**Example: PUT area/1/set**

```
{
   "status":"success",
   "data": "null"
}
```

## 3.4 Zone Resources

### 3.4.1 Zone Requests

| Method | Resource | Description |
|---|---|---|

| GET | zone | Returns the status of all zones |
|---|---|---|
| GET | zone/\<ID\> | Returns the status of zone \<ID\>. |
| GET | zone/\<ELEMENT\> | Returns \<ELEMENT\> value of all zones. |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ID | The zone id. (1 – number of defined zones) |
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

There will be one instance of these elements for each zone.

| Element | Type | Description |
|---|---|---|
| id | Integer | Zone ID (1 to maximum number of zones) |
| type | Integer | Zone type, as follows:<br>0 - Alarm<br>1 - Entry/Exit<br>2 - Exit Terminator<br>3 - Fire<br>4 - Fire Exit<br>5 - Line<br>6 - Panic<br>7 - Hold-up<br>8 - Tamper<br>9 - Technical<br>10 - Medical<br>11 - Keyarm<br>12 – Unused<br>13 - Shunt<br>14 - X-shunt<br>15 - Fault<br>16 - Lock Supervision<br>17 - Seismic<br>18 - All Okay |
| zone_name | String | Zone name |
| area | Integer | Area number (1 – max number of areas) |
| area_name | String | Area name |
| input | Integer | Zone input as follows:<br>0 - Closed<br>1 - Open<br>2 - Short<br>3 - Disconnected<br>4 - PIR Masked<br>5 - DC Substitution<br>6 - Sensor Missing<br>7 - Offline |
| status | Integer | Zone status as follows:<br>0 - OK<br>1 - Inhibit<br>2 - Isolate<br>3 - Soak |

| | | 4 - Tamper<br>5 - Alarm<br>6 – (OK but not used recently)<br>7 - Trouble |
|---|---|---|

## Example: GET zone

```
{
    "status": "success",
    "data": {
        "zone": [
            {
                "id": "1",
                "type": "0",
                "zone_name": "Entrance",
                "area": "1",
                "area_name": "Area 1",
                "input": "0",
                "status": "0"
            },
            {
                "id": "2",
                "type": "0",
                "zone_name": "Kitchen",
                "area": "1",
                "area_name": "Area 1",
                "input": "1",
                "status": "0"
            },
            {
                "id": "3",
                "type": "0",
                "zone_name": "Living Room",
                "area": "1",
                "area_name": "Area 1",
                "input": "3",
                "status": "2"
            }
        ]
    }
}
```

## Example: GET zone/1

```
{
    "status": "success",
    "data": {
        "zone": [
            {
                "id": "1",
                "type": "0",
                "zone_name": "Entrance",
                "area": "1",
                "area_name": "Area 1",
                "input": "0",
                "status": "0"
```

```
                }
            ]
        }
}
```

### Example: GET zone/input

```
{
        "status": "success",
        "data": {
            "zone": [
                {
                        "input": "0"
                },
                {
                        "input": "1"
                },
                {
                        "input": "3"
                }
            ]
        }
}
```

### 3.4.2   Zone Commands

| Method | Resource | Description |
| --- | --- | --- |
| PUT | zone/<ID>/inhibit | Inhibit zone <ID> |
| PUT | zone/<ID>/deinhibit | Deinhibit zone <ID> |
| PUT | zone/<ID>/isolate | Isolate zone <ID> |
| PUT | zone/<ID>/deisolate | Deisolate zone <ID> |

Resource Parameters

| Element | Description |
| --- | --- |
| ID | The zone id. (1 – number of defined zones) |

### Example: PUT zone/1/inhibit

```
{
   "status":"success",
   "data": "null"
}
```

## 3.5 Output Resources

### 3.5.1 Output Requests

| Method | Resource | Description |
|---|---|---|
| GET | output | Returns the status of all outputs |
| GET | output/<ID> | Returns the status of output <ID>. |
| GET | output/<ELEMENT> | Returns <ELEMENT> value of all outputs. |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ID | The output id. (1 – number of defined outputs) |
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

There will be one instance of these elements for each output.

| Element | Type | Description |
|---|---|---|
| id | Integer | Output ID (1 to maximum number of outputs) |
| name | String | Name of output |
| state | Integer | Output state as follows:<br>0 – Not set<br>1 - Set |

**Example: GET output**

```
{
      "status": "success",
      "data": {
            "output": [
                  {
                        "id": "1",
                        "name": "Output 1",
                        "state": "0"
                  },
                  {
                        "id": "2",
                        "name": "Output 2",
                        "state": "1"
                  }
            ]
      }
}
```

**Example: GET output/state**

```
{
      "status": "success",
      "data": {
```

```
        "output": [
                {
                        "state": "0"
                },
                {
                        "state": "1"
                }
        ]
    }
}
```

### 3.5.2   Output Commands

| Method | Resource | Description |
| --- | --- | --- |
| PUT | output/<ID>/set | Set output <ID> |
| PUT | output/<ID>/reset | Reset output <ID> |

Resource Parameters

| Element | Description |
| --- | --- |
| ID | The output id. (1 – number of defined output) |

**Example: PUT output/1/set**

```
{
   "status":"success",
   "data": "null"
}
```

## 3.6   Bell Resources

### 3.6.1   Bell Commands

| Method | Resource | Description |
| --- | --- | --- |
| PUT | bell/silence | Silence all bells |

**Example: PUT bell/silence**

```
{
   "status":"success",
   "data": "null"
}
```

## 3.7 Basic Panel Info Resources

### 3.7.1 Basic Panel Info Requests

| Method | Resource | Description |
|---|---|---|
| GET | panel | Returns basic panel info |
| GET | panel/<ELEMENT> | Returns <ELEMENT> value |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

| Element | Type | Description |
|---|---|---|
| type | String | Panel type |
| variant | String | Panel subtype |
| version | String | Firmware version of panel |
| device-id | Integer | Installation ID |
| sn | Hexadecimal | Serial number of panel |
| cfgtime | Timestamp | Timestamp of last configuration change. Present only if available. |
| hw_ver_major | Integer | Major hardware version of panel |
| hw_ver_minor | Integer | Minor hardware version of panel |
| license_key | String | Encrypted license key, presented as a string encoded in Base-32. |

**Example: GET panel**

```
{
     "status": "success",
     "data": {
          "panel": {
               "type": "SPC4000",
               "variant": "4300",
               "version": "3.2.5",
               "device-id": "1",
               "sn": "123ABC99",
               "cfgtime": "1390248400",
               "hw_ver_major": "1",
               "hw_ver_minor": "4",
               "license_key": "XXXXXXXXXXXXX"
          }
     }
}
```

**Example: GET panel/version**

```
{
     "status": "success",
```

```
    "data": {
        "panel": {
            "version": "3.2.5"
        }
    }
}
```

## 3.8   System Info Resources

### 3.8.1   System Info Requests

| Method | Resource | Description |
| --- | --- | --- |
| GET | system | Returns system info |
| GET | system/<ELEMENT> | Returns <ELEMENT> value |

**Resource Parameters**

| Parameter | Description |
| --- | --- |
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

| Element | Type | Description |
| --- | --- | --- |
| time | Timestamp | Panel time |
| engmode | Integer | Engineer mode:<br>0 – no<br>1 - yes |
| rf_type | Integer | Type of wireless module fitted on panel, as follows:<br>0 - Not fitted<br>1 - Visonic<br>2 - SiWay |
| rf_version | Integer | Wireless module firmware version |

**Example: GET system**

```
{
    "status": "success",
    "data": {
        "system": {
            "time": "1390635628",
            "engmode": "0",
            "rf_type": "2",
            "rf_version": "10"
        }
    }
}
```

## 3.9 Power Supply Resources

### 3.9.1 Power Supply Requests

| Method | Resource | Description |
|---|---|---|
| GET | psu | Returns power supply info |
| GET | psu/<ELEMENT> | Returns <ELEMENT> value |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

| Element | Type | Description |
|---|---|---|
| batt_volt | String | Battery voltage |
| aux_volt | String | Voltage at auxiliary output |
| aux_curr | String | Current drawn from auxiliary output |
| ac_freq | String | Main frequency |

**Example: GET psu**

```
{
      "status": "success",
      "data": {
            "psu": {
                  "batt_volt": "9.8V",
                  "aux_volt": "13.4V",
                  "aux_curr": "120mA",
                  "ac_freq": "50Hz"
            }
      }
}
```

## 3.10 System Alerts Resources

### 3.10.1 System Alerts Requests

| Method | Resource | Description |
|---|---|---|
| GET | alert | Returns system alerts |
| GET | alert/<ELEMENT> | Returns <ELEMENT> value |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ELEMENT | See valid element names in Return Values table below |

## System alerts bit definition

The system alert states are presented as 8-digit hexadecimal bitmasks, and all have the following bit assignments:

| Bit | Description |
|---|---|
| 0 | Mains power fault (running on battery) |
| 1 | Battery fault |
| 2 | Auxiliary fuse |
| 3 | External bell fuse |
| 4 | Internal bell fuse |
| 5 | Bell tamper input |
| 6 | Cabinet Tamper switch open |
| 7 | Auxiliary Tamper 1 |
| 8 | Auxiliary Tamper 2 |
| 9 | Wireless antenna tamper detected |
| 10 | Wireless jamming detected |
| 11 | Modem 1 fault state |
| 12 | Modem 1 phone line fault |
| 13 | Modem 2 fault state |
| 14 | Modem 2 phone line fault |
| 15 | Cable fault |
| 16 | Fail to communicate |
| 17 | User duress |
| 18 | Date/Time lost (SPC 4000 only) |
| 19 | User RF Panic button pressed |
| 20 | User Man Down Transmitter alarm |
| 21 | Panel power supply problem |

## Return Values

| Element | Type | Description |
|---|---|---|
| input | Hexadecimal | Input state per the bit definition above |
| alert | Hexadecimal | Alert state per the bit definition above |
| inhibit | Hexadecimal | Inhibit state per the bit definition above |
| isolate | Hexadecimal | Isolate state per the bit definition above |

### Example: GET alert

```
{
      "status": "success",
      "data": {
            "alert": {
                  "input": "00000002",
                  "alert": "00000000",
```

```
                      "inhibit": "00000000",
                      "isolate": "00000002"
                }
          }
}
```

### 3.10.2 System Alerts Commands

| Method | Resource | Description |
|---|---|---|
| PUT | alert/restore | Restores all alerts. |

**Example: PUT alert/restore**

```
{
   "status":"success",
   "data": "null"
}
```

## 3.11 Modem Info Resources

### 3.11.1 Modem Info Requests

| Method | Resource | Description |
|---|---|---|
| GET | modem | Returns modem info |
| GET | modem/<ELEMENT> | Returns <ELEMENT> value of all modems |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

There will be one instance of these elements for each fitted modem (up to two).

| Element | Type | Description |
|---|---|---|
| port | Integer | Modem ID: 1 – primary, 2 - backup |
| enabled | Integer | Modem enabled: 0 – no, 1 - yes |
| status | Integer | Modem status as follows:<br>0 - Disabled<br>1 - OK<br>2 - No Response<br>3 - Fault<br>4 - Bad PIN<br>5 - Line Fault<br>6 - Bad SIM<br>7 - Unsupported country code |

| state | Integer | Modem state as follows:<br>0 - Not Applicable(because modem is disabled)<br>1 - Ready<br>2 - Detecting<br>3 - Initializing<br>4 - Dialling<br>5 - Answering<br>6 - Connected<br>7 - Disconnecting<br>8 - Busy<br>9 - GPRS (dial, connected or disconnecting) |
| --- | --- | --- |
| type | Integer | Type of modem fitted as follows:<br>0 - None<br>1 - PSTN<br>2 - GSM<br>3 - Redcare |
| id_type | String | Modem type ID. |
| id_fw | String | Modem firmware ID |
| id_hw | String | Modem hardware ID |
| capabilities | Hexadecimal | Modem capabilities is a bitmask, as follows:<br>Bit 0 - slow dial<br>Bit 1 - fast dial<br>Bit 2 - SMS<br>Bit 3 - Fast Format<br>Bit 4 - Contact ID<br>Bit 5 - SIA<br>Bit 6 - SMS Incoming<br>Bit 7 - GPRS<br>Bit 8 - Multiple SIA events in a single call.<br>Bit 9 - Multiple ContactID events in a single call |
| gsm_signal | Integer | GSM signal strength, applicable only to GSM modems. The range is 2 – 9, 2 is lowest. |
| status_raw | Hexadecimal | Raw modem status. Bitmask as follows:<br>Bit 0 - modem is detected<br>Bit 1 - modem is not responding (fault)<br>Bit 2 - line fault |
| last_call | Timestamp | Time of last call. Present only if available |
| last_direction | Integer | Direction of last call: 0 – in, 1 – out. Present only if available |
| incoming_time | Integer | Total time connected for incoming calls, in seconds. |
| incoming_count | Integer | Number of incoming calls |
| outgoing_time | Integer | Total time connected for outgoing calls, in seconds. |
| outgoing_count | Integer | Number of outgoing calls. |
| outgoing_failed | Integer | Number of failed dial attempts. |
| incoming_sms_count | Integer | Number of received SMS. |
| outgoing_sms_count | Integer | Number of sent SMS. |

## Example: GET modem

```
{
      "status": "success",
      "data": {
            "modem": [
                  {
                        "port": "1",
                        "enabled": "1",
                        "status": "1",
```

```
                    "state": "1",
                    "type": "2",
                    "id_type": "IntelliModem GSM",
                    "id_fw": "3.07 [28MAY12]",
                    "id_hw": "2",
                    "capabilities": "03FF",
                    "gsm_signal": "3",
                    "status_raw": "0001",
                    "incoming_time": "0",
                    "incoming_count": "0",
                    "outgoing_time": "0",
                    "outgoing_count": "0",
                    "outgoing_failed": "0",
                    "incoming_sms_count": "2",
                    "outgoing_sms_count": "0"
                },
                {
                    "port": "2",
                    "enabled": "0",
                    "status": "0",
                    "state": "0",
                    "type": "1",
                    "id_type": "",
                    "id_fw": "0.00 []",
                    "id_hw": "",
                    "capabilities": "0000",
                    "gsm_signal": "0",
                    "status_raw": "0000",
                    "incoming_time": "0",
                    "incoming_count": "0",
                    "outgoing_time": "0",
                    "outgoing_count": "0",
                    "outgoing_failed": "0",
                    "incoming_sms_count": "0",
                    "outgoing_sms_count": "0"
                }
            ]
        }
}
```

## 3.12 Ethernet Info Resources

### 3.12.1 Ethernet Info Requests

| Method | Resource | Description |
|---|---|---|
| GET | ethernet | Returns ethernet interface info |
| GET | ethernet/<ELEMENT> | Returns <ELEMENT> value |

**Resource Parameters**

| Parameter | Description |
|---|---|
| | |

| ELEMENT | See valid element names in Return Values table below |
|---|---|

## Return Values

| Element | Type | Description |
|---|---|---|
| fitted | Integer | Ethernet circuit fitted. 0 – no, 1 - yes |
| state | Integer | Ethernet state. 0 - down, 1 - up |
| dhcp_enabled | Integer | DHCP enabled. 0 – no, 1 - yes |
| mac_address | String | MAC address |
| ip_address | String | IP address |
| netmask | String | Netmask |
| gateway | String | Gateway IP address |
| tx_packets | Integer | Count of packets transmitted |
| tx_bytes | Integer | Count of octets transmitted |
| rx_packets | Integer | Count of packets received |
| rx_bytes | Integer | Count of octets received |

## Example: GET ethernet

```
{
      "status": "success",
      "data": {
            "ethernet": {
                  "fitted": "1",
                  "state": "1",
                  "dhcp_enabled": "0",
                  "mac_address": "FF:FF:FF:FF:FF:FF",
                  "ip_address": "192.168.0.2",
                  "netmask": "255.255.255.0",
                  "gateway": "192.168.0.1",
                  "tx_packets": "348988889",
                  "tx_bytes": "2061242289",
                  "rx_packets": "392464246",
                  "rx_bytes": "2417087670"
            }
      }
}
```

## 3.13 User Info Resources

### 3.13.1 User Info Requests

| Method | Resource | Description |
|---|---|---|
| GET | user | Returns info about all users |
| GET | user/<ID> | Returns info about user <ID> |
| GET | user/<ELEMENT> | Returns <ELEMENT> value of all users |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ID | User ID |
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

There will be one instance of these elements for each user.

| Element | Type | Description |
|---|---|---|
| id | Integer | User ID |
| name | String | User name |

**Example: GET user**

```
{
      "status": "success",
      "data": {
           "user": [
                {
                       "id": "1",
                       "name": "User 1"
                },
                {
                       "id": "2",
                       "name": "User 2"
                },
                {
                       "id": "3",
                       "name": "User 3"
                }
           ]
      }
}
```

## 3.14 X-BUS Node Info Resources

### 3.14.1 X-BUS Node Info Requests

| Method | Resource | Description |
|---|---|---|
| GET | xbusnode | Returns info about all X-BUS nodes |
| GET | xbusnode/<ELEMENT> | Returns <ELEMENT> value of all X-BUS-nodes |

**Resource Parameters**

| Parameter | Description |
|---|---|
| | |

| ELEMENT | See valid element names in Return Values table below |
| --- | --- |

## Return Values

There will be one instance of these elements for each connected X-BUS nodes.

| Element | Type | Description |
| --- | --- | --- |
| id | Integer | X-BUS node ID |
| sn | Hexadecimal | Serial number |
| name | String | Name of the node |
| type | Integer | Node type as follows:<br>1 - Standard Keypad<br>2 - Standard I/O<br>3 - Analysed I/O<br>4 - IntelliPower PSU Interface (hardware version 1 only)<br>5 - Wireless (hardware version 1 only)<br>6 - Two reader door controller<br>7 - Multi Area Keypad<br>8 - Indicator<br>9 - Key Switch<br>10 - Audio Expander |
| hardware_id | Integer | Hardware version:<br>1 - Original version.<br>2 - Version with rotary switches. |
| icount | Integer | Number of physical inputs |
| ocount | Integer | Number of physical outputs |
| version | String | Firmware version |
| rf_type | Integer | Type of wireless module piggybacked on a node of hardware version 2, as follows:<br>0 - Not fitted<br>1 - Visonic<br>2 - SiWay |
| rf_version | Integer | Firmware version of wireless module. |
| reader_type | Integer | Type of card reader, as follows (this applies only to keypads):<br>0 - None fitted<br>1 - EM4100<br>2 - MIFARE |
| status | Hexadecimal | Node status. Bitmask, as follows:<br>Bit 0 - Node was reset<br>Bit 1 - Node 12V supply is low<br>Bit 2 - Node Lid is open<br>Bit 3 - Node Back tamper is open<br>Bit 4 - Battery fault<br>Bit 5 - Mains fault<br>Bit 6 - Over current<br>Bit 7 - RF Jamming detected<br>Bit 8 - External antenna not connected.<br>Bit 9 - Node fuse 1 blown<br>Bit 10 - Node fuse 2 blown<br>Bit 11 - Node fuse 3 blown<br>Bit 12 - Node fuse 4 blown<br>Bit 13 - PSU fault |
| position_1 | Integer | Node position on cable 1 |
| position_2 | Integer | Node position on cable 2. |
| psu_type | Integer | Type of PSU. This applies to hardware of version 2 and greater, as follows:<br>0 - No PSU is attached to this node. |

| | | 1 - PSU attached (3 A) |
|---|---|---|
| psu_fw_version | Integer | Firmware version of PSU attached to this node. Applicable only if PSU_TYPE is not zero |
| psu_mode | Integer | Battery mode, as follows:<br>1 - no battery jumper fitted (fault condition)<br>2 - battery jumper set for output voltage limited to 12 V (that is, battery will not be charged)<br>3 - battery jumper set for 7 Ah battery<br>4 - battery jumper set for 17 Ah battery |
| psu_out1_volt | String | Voltage at PSU output 1. Present only if PSU_TYPE is not zero. |
| psu_out1_curr | String | Current at PSU output 1. Present only if PSU_TYPE is not zero. |
| psu_out2_volt | String | Voltage at PSU output 2. Present only if PSU_TYPE is not zero. |
| psu_out2_curr | String | Current at PSU output 2. Present only if PSU_TYPE is not zero. |
| psu_out3_volt | String | Voltage at PSU output 3. Present only if PSU_TYPE is not zero. |
| psu_out3_curr | String | Current at PSU output 3. Present only if PSU_TYPE is not zero. |
| psu_batt_volt | String | Battery current. Present only if PSU_TYPE is not zero or TYPE is 4. |
| psu_batt_curr | String | Current at PSU output 1. Present only if PSU_TYPE is not zero or TYPE is 4. |
| aux_volt | Sting | Voltage at auxiliary output (to power relays, for example). For TYPE 4 (IntelliPower PSU) this is the voltage supplied. |
| aux_curr | String | Current at auxiliary output. For TYPE 4 (IntelliPower PSU) this is sum of current supplied on outputs 1 and 2 |
| input | Hexadecimal | Input state. Bitmask as follows:<br>Bit 0 - Configured node is no longer communicating<br>Bit 1 - Node cabinet tamper<br>Bit 2 - Node fuse fault<br>Bit 3 - Node Mains fault<br>Bit 4 - Node Battery fault<br>Bit 5 - Node has detected RF jamming<br>Bit 6 - Node has detected antenna tamper<br>Bit 7 - Node Panic button is pressed<br>Bit 8 - Node Fire button is pressed<br>Bit 9 - Node Medical button is pressed<br>Bit 10 - Node PSU fault |
| alert | Hexadecimal | Alert state per the description above |
| inhibit | Hexadecimal | Inhibit state per the description above. |
| isolate | Hexadecimal | Isolate state per the description above. |
| key_position | Integer | Position of key, present only for nodes of TYPE 9 (Key Switch), as follows:<br>0 - Centre position<br>1 - Right position<br>2 - Left position<br>254 - Tamper<br>255 - Unknown position |
| audio_type | Integer | Audio type as follows:<br>0 - Not fitted<br>1 - Capture and playback<br>Present only for nodes of type Multi Area Keypad |

**Example: GET xbusnode**

```
{
     "status": "success",
     "data": {
          "xbusnode": [
                {
```

```
                    "id": "1",
                    "sn": "9999999",
                    "name": "Extra I/O",
                    "type": "2",
                    "hardware_id": "1",
                    "icount": "8",
                    "ocount": "2",
                    "version": "1.09 13DEC10",
                    "rf_type": "0",
                    "rf_version": "0",
                    "reader_type": "0",
                    "status": "00000000",
                    "position_1": "2",
                    "position_2": "0",
                    "psu_type": "0",
                    "aux_volt": "13.9V",
                    "aux_curr": "47mA",
                    "input": "0000",
                    "alert": "0000",
                    "inhibit": "0000",
                    "isolate": "0000"
                },
                {
                    "id": "2",
                    "sn": "FFFFFFFF",
                    "name": "Keypad 1",
                    "type": "1",
                    "hardware_id": "1",
                    "icount": "0",
                    "ocount": "0",
                    "version": "2.09 13MAR13",
                    "rf_type": "0",
                    "rf_version": "0",
                    "reader_type": "1",
                    "status": "00000004",
                    "position_1": "1",
                    "position_2": "0",
                    "psu_type": "0",
                    "aux_volt": "13.6V",
                    "aux_curr": "0mA",
                    "input": "0002",
                    "alert": "0000",
                    "inhibit": "0000",
                    "isolate": "0002"
                }
            ]
        }
}
```

## 3.15 X-BUS Map Info Resources

### 3.15.1 X-BUS Map Info Requests

| Method | Resource | Description |
|---|---|---|
| GET | xbusmap | Returns info about the X-BUS map |
| GET | xbusmap/<ELEMENT> | Returns <ELEMENT> value of all X-BUS-nodes |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

There will be two instances of these elements, one for each connected X-BUS port.

| Element | Type | Description |
|---|---|---|
| port | Integer | X-BUS port, can be 1 or 2. |
| node_count | Integer | How many nodes connected on this port. In a ring configuration, this attribute will be the same for both ports. |
| topology | Integer | X-BUS topology. 0 - chain or ring, 1 - multi-drop |

There will be one instance of these elements for each connected X-BUS node per X-BUS port.

| Element | Type | Description |
|---|---|---|
| position | Integer | In a chain or ring configuration, this indicates the physical placing on the network. Counting starts at the panel that is assigned the 0 position. In a multi-drop configuration this attribute is assigned by the system but has otherwise no other meaning |
| switch_id | Integer | Rotary switch setting if the node has it, otherwise will be set to 255. |
| sn | Hexadecimal | Serial number |
| type | Integer | Node type as follows:<br>1 - Standard Keypad<br>2 - Standard I/O<br>3 - Analysed I/O<br>4 - IntelliPower PSU Interface<br>5 - Wireless<br>6 - Two reader door controller<br>7 - Multi Area Keypad<br>8 - Indicator<br>9 - Key Switch<br>10 - Audio Expander |
| hardware_id | Integer | Hardware version:<br>1: Original version.<br>2: Version with rotary switches. |
| icount | Integer | Number of inputs |
| ocount | Integer | Number of outputs |

| version | String | Firmware version |
|---|---|---|
| licensed | Integer | Node is licensed for use with the panel it is connected to. 0 – no, 1 - yes |

**Example: GET xbusmap**

```
{
    "status": "success",
    "data": {
        "xbusmap": [
            {
                "port": "1",
                "node_count": "2",
                "topology": "1",
                "node": [
                    {
                        "position": "1",
                        "switch_id": "2",
                        "sn": "9999999",
                        "type": "1",
                        "hardware_id": "1",
                        "icount": "0",
                        "ocount": "0",
                        "version": "2.09 13MAR13",
                        "licensed": "1"
                    },
                    {
                        "position": "2",
                        "switch_id": "1",
                        "sn": "FFFFFFF",
                        "type": "2",
                        "hardware_id": "1",
                        "icount": "8",
                        "ocount": "2",
                        "version": "1.09 13DEC10",
                        "licensed": "1"
                    }
                ]
            },
            {
                "port": "2",
                "node_count": "0",
                "topology": "0"
            }
        ]
    }
}
```

## 3.16 Door Resources

### 3.16.1 Door Requests

| Method | Resource | Description |
|---|---|---|
| GET | door | Returns the status of all doors |
| GET | door/<ID> | Returns the status of door <ID> |
| GET | door/<ELEMENT> | Returns <ELEMENT> value of all doors |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ID | Door ID |
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

There will be one instance of these elements for each configured door.

| Element | Type | Description |
|---|---|---|
| id | Integer | Door ID, from 1 to max number of doors |
| dps_input | Integer | DPS zone input as follows:<br>0 - Closed<br>1 - Open<br>2 - Short<br>3 - Disconnect<br>4 - N/A<br>5 - DC Substitution<br>6 - N/A<br>7 - Offline |
| drs_input | Integer | DRS zone input as follows:<br>0 - Closed<br>1 - Open<br>2 - Short<br>3 - Disconnect<br>4 - N/A<br>5 - DC Substitution<br>6 - N/A<br>7 - Offline |
| status | Integer | Door status as follows:<br>0 - OK<br>1 - Open too long<br>2 - Left open<br>3 - Forced<br>4 - Tamper<br>5 - Offline |
| mode | Integer | Door mode. 0 – Normal, 1- Locked, 2 - Blocked |
| zone | Integer | Zone ID corresponding to this door, from 1 to maximum number of zones. Present only if available. |
| zone_name | String | Zone name. Present only if available |
| area | Integer | Area number, from 1 to maximum number of areas.<br>Present only if available |

| area_name | String | Area name. Present only if available |
|---|---|---|

**Example: GET door**

```
{
    "status": "success",
    "data": {
        "door": [
            {
                "id": "1",
                "dps_input": "1",
                "drs_input": "0",
                "status": "0",
                "mode": "0",
                "zone": "5",
                "zone_name": "Door 1",
                "area": "1",
                "area_name": "Area 1"
            },
            {
                id": "2",
                "dps_input": "1",
                "drs_input": "0",
                "status": "0",
                "mode": "1"
            }
        ]
    }
}
```

## 3.17 Verification Resources

### 3.17.1 Verification Zone Requests

| Method | Resource | Description |
|---|---|---|
| GET | vzone | Returns the status of all verification zones |
| GET | vzone/<ID> | Returns the status of verification zone <ID> |
| GET | vzone/<ELEMENT> | Returns <ELEMENT> value of all verification zones |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ID | Verification zone ID |
| ELEMENT | See valid element names in Return Values table below |

**Return Values**

There will be one instance of these elements for each verification zone.

| Element | Type | Description |
|---|---|---|
| id | Integer | Verification zone ID, from 1 to max number of verification zones |
| name | String | Name of verification zone |
| audio | Integer | Audio state. 0 - No audio available, 1 - Audio online, 2 - Audio offline |
| audio_pre | Integer | Duration of pre-event recording, in seconds. Only present if a verification event is available and pre-event audio recording is enabled. |
| audio_post | Integer | Duration of post-event recording, in seconds. Only present if a verification event is available and post-event audio recording is enabled. |
| audio_volume | Integer | Volume for speaker, ranging from 0 to 7, where 0 means the speaker is disabled, 1 is minimum volume, 7 is maximum. Only present if audio is online. |
| video | Integer | Video status. 0 - No video available, 1 - Video online, 2 - Video offline |
| pre_images | Integer | Number of recorded prevent images. Only present if a verification event is available and pre-event video recording is enabled. |
| pre_interval | Integer | Interval between pre-event images, in seconds (1 -10). Only present if a verification event is available and pre-event video recording is enabled |
| post_images | Integer | Number of recorded postevent images. Only present if a verification event is available and post-event video recording is enabled. |
| post_interval | Integer | Interval between postevent images, in seconds (1 - 10). Only present if a verification event is available and post-event video recording is enabled. |
| vtime | Timestamp | Timestamp of the moment a verification event was triggered for this zone. This is normally the same as the SIA event timestamp, however for an entry event the timestamp will be that of the entry timer expiring, whereas this verification timestamp will refer to the moment of the opening of the entry zone. Present only if available. (FW3.1 and greater) |
| last_open | Timestamp | Timestamp of last time a physical zone assigned to this verification zone was open. This can be used for example to help with tracking movement inside a building that has PIRs, as zones open and close when a burglar moves around, and a second event for a zone will not be issued if the alarm for the first event has not been restored. Present only if available (a zone has opened). |

## Example: GET verification

```
{
     "status": "success",
     "data": {
          "vzone": [
               {
                    "id": "1",
                    "name": "Vzone 1",
                    "audio": "0",
                    "video": "1"
               }
          ]
     }
}
```

## 3.18 Image Resources

### 3.18.1 Image Requests

| Method | Resource | Description |
|--------|----------|-------------|
| GET | image/<VZONE_ID> | Returns a live image for verification zone <VZONE_ID> |
| GET | image/<VZONE_ID>/<IMAGE_ID> | Returns recorded image <IMAGE_ID> for verification zone <VZONE_ID>. |

**Resource Parameters**

| Parameter | Description |
|-----------|-------------|
| VZONE_ID | Verification zone  ID |
| IMAGE_ID | ID -1 to -16 returns image saved before verification zone event.<br>ID 1 to 16 returns image saved after verification zone event.<br>IDs are ordered in time as follows: -16, -15, … ,-2,-1, (event), 1, 2, … , 15, 16.<br>Image ID 0 will return the live image. |
| | |

**Return Values**

| Element | Type | Description |
|---------|------|-------------|
| id | Integer | Image ID:<br> -16 to -1 – Pre-event image<br>0 – Live image<br>1 to 16 – Post-event image |
| timestamp_s | Timestamp | Time when image was captured (seconds) |
| timestamp_ms | Integer | Fraction of a second for the timestamp, in milliseconds (10 milliseconds precision). |
| format | String | Image format. Currently only "jpeg/base64" is supported. |
| data | String | String containing base64 encoded jpeg image data |

**Example: GET image/1 or image/1/0**

```
{
      "status": "success",
      "data": {
            "image": {
                        "id": "0"
                        "timestamp_s": "1390641422",
                        "timestamp_ms":"250",
                        "format": "jpeg/base64",
                        "data": "…"
            }
      }
}
```

**Example: GET image/1/-1**

```
{
      "status": "success",
```

```
      "data": {
            "image": {
                        "image_id": "-1"
                        "image_timestamp_s": "1390641301",
                        "image_timestamp_ms":"120",
                        "format": "jpeg/base64",
                        "data": "…"
                  }
            }
}
```

## 3.19 System Log Resources

### 3.19.1 System Log Requests

| Method | Resource | Description |
|--------|----------|-------------|
| GET | systemlog | Returns systemlog events  (last 100 events) |
| GET | systemlog/<MAX_NUM_EVENTS> | Returns last <MAX_NUM_EVENTS>. |

**Resource Parameters**

| Parameter | Description |
|-----------|-------------|
| MAX_NUM_EVENTS | Number of requested events. 1 to 500. |

**Return Values**

There will be one instance of these elements for each **event.**

| Element | Type | Description |
|---------|------|-------------|
| time | Timestamp | Event time |
| raw | Hexadecimal | Event raw data. Its encoding is not specified in this document. |
| text | String | Textual description of event. |

**Example: GET systemlog**

```
{
      "status": "success",
      "data": {
            "systemlog": {
                  "event": [
                        {
                              "time": "1390642515",
                              "raw": "0800010000000000",
                              "text": "EVENT TEXT 1"
                        },
                        {
                              "time": "1390642159",
                              "raw": "0800010001000000",
```

```
                        "text": "EVENT TEXT 2"
                    },

                    …

                    {
                            "time": "1390642515",
                            "raw": "0800010000000000",
                            "text": "EVENT TEXT 100"
                    }
                ]
            }
        }
}
```

## 3.20 Access Log Resources

### 3.20.1 Access Log Requests

| Method | Resource | Description |
|---|---|---|
| GET | accesslog | Returns accesslog events  (last 100 events) |
| GET | accesslog/<MAX_NUM_EVENTS> | Returns last <MAX_NUM_EVENTS> |

**Resource Parameters**

| Parameter | Description |
|---|---|
| MAX_NUM_EVENTS | Number of requested events. 1 to 500. |

**Return Values**

There will be one instance of these elements for each **event.**

| Element | Type | Description |
|---|---|---|
| time | Timestamp | Event time |
| type | Integer | Event type, as follows:<br>1 - Entry granted<br>2 - Entry denied<br>3 - Exit granted<br>4 - Exit denied<br>5 - Door release (the door release button is pressed while the door is in normal mode)<br>6 - Door open too long<br>7 - Door left open<br>8 - Door forced open<br>9 - Door normal<br>10 - Door locked<br>11 - Door unlocked<br>12 - Passback violation<br>13 - Card voided (as a Comment result of entering too many invalid PIN entries)<br>14 - Card added |

| | | 15 - Card deleted<br>16 - Card reset<br>17 - Card details change<br>18 - Card data<br>19 - Card voided at the keypad |
|---|---|---|
| card_format | Integer | Card type, present only for events of type 18 (Card Data):<br>0 - Wiegand 26 bits<br>1 - Wiegand 36 bits (EPX2)<br>2 - HID Corporate 1000<br>3 - EM4102<br>4 – COTAG |
| card_site | Integer | Site number for cards with CARD_FORMAT equal to 0, 1, or 2, otherwise zero. Present only for events of type 18 (Card Data). |
| card_number | Integer | Card number for cards with CARD_FORMAT equal to 0, 1, or 2, otherwise zero. Present only for events of type 18 (Card Data). Due to implementation reasons (flash structure) this number is wrong and should not be relied upon. |
| card_number_raw | String | Raw card number in the internal panel format, presented as "<high> <low>", where <high> and <low> are both decimal numbers. Present only for events of type 18 (Card Data). Due to implementation reasons (flash structure) <low> is wrong and should not be relied upon. |
| detail | Integer | Event detail, present only if available (not zero). Currently this attribute provides additional detail only for entry denied events, as follows:<br>1 - Passback violation<br>2 - Escort required<br>3 - Custodian required<br>4 - Invalid PIN<br>5 - Area is fullset<br>6 - Door is locked<br>7 - Invalid card format<br>8 - Card not on system<br>9 - Door not authorized for this card<br>10 - Outside time/date limit<br>11 - Card is void<br>12 - Interlocked door is open |
| door_id | Integer | Door ID for this event,present only if available (not zero). |
| door_name | String | Zone name for the door. This is present only if there a zone associated with the door and if the zone has a name defined. |
| user_changed_id | Integer | ID of user for which a card was changed. Present only for events of type 14, 15, 16, 17 and 19 if there is an associated user. (For those type of events USER_ID, defined below, indicates who performed the card operation.) |
| user_changed_name | String | Name of user for which a card was changed. Present only for events of type 14, 15, 16, 17 and 19 if there is an associated user and it has a name defined. (For those type of events USER_NAME, defined below, indicates who performed the card operation.) |
| user_id | Integer | User ID for this event. This attribute is present only if there is an associated user. |
| user_name | String | User Name for this event. This attribute is present only if there is an associated user and it has a name defined |

### Example: GET accesslog

```
{
     "status": "success",
     "data": {
          "accesslog": {
```

```
                "event": [
                    {
                        "time": "1390642515",
                        "type": "1",
                        "door_id": "1",
                        "door_name": "Door 1",
                        "user_id": "1",
                        "user_name": "User 1"
                    },
                    {
                        "time": "1390642515",
                        "type": "2",
                        "door_id": "1",
                        "door_name": "Door 1",
                        "user_id": "3",
                        "user_name": "User 3"
                    }
                ]
            }
        }
}
```

## 3.21 Zone Log Resources

### 3.21.1 Zone Log Requests

| Method | Resource | Description |
|---|---|---|
| GET | zonelog/<ZONE_ID> | Returns zonelog events for zone <ZONE_ID> |

**Resource Parameters**

| Parameter | Description |
|---|---|
| ZONE_ID | Zone ID. 1 to maximum number of zones. |

**Return Values**

| Element | Type | Description |
|---|---|---|
| zone | Integer | Zone ID |

There will be one instance of these elements for each **event.**

| Element | Type | Description |
|---|---|---|
| time | Timestamp | Event time |
| input | Integer | Zone state as follows:<br>0 - Closed<br>1 - Open<br>2 - Short<br>3 - Disconnected<br>4 - PIR Masked |

| | | 5 - DC Substitution |
| | | 6 - N/A |
| | | 7 - Offline |
| | | 8 - Pulse Count |
| | | 9 -Gross Attack |

**Example: GET zonelog/8**

```
{
      "status": "success",
      "data": {
          "zonelog": {
                "zone": "8",
                "event": [
                        {
                                "time": "1389030757",
                                "input": "1"
                        },
                        {
                                "time": "1389030761",
                                "input": "0"
                        },
                        {
                                "time": "1389032060",
                                "input": "1"
                        },
                        {
                                "time": "1389032064",
                                "input": "0"
                        }
                ]
          }
      }
}
```

## 3.22 Wireless Log Resources

### 3.22.1 Wireless Log Requests

| Method | Resource | Description |
|--------|----------|-------------|
| GET | wirelesslog/<SENSOR_ID> | Returns wirelesslog events for sensor <SENSOR_ID> |

**Resource Parameters**

| Parameter | Description |
|-----------|-------------|
| SENSOR_ID | Sensor ID |

## Return Values

| Element | Type | Description |
|---|---|---|
| sensor | Integer | Sensor ID |

There will be one instance of these elements for each **event.**

| Element | Type | Description |
|---|---|---|
| time | Timestamp | Event time |
| status | Integer | Sensor status as follows:<br>0 - Closed<br>1 - Tamper<br>2 - Open<br>3 - Fault |
| battery | Integer | Battery status as follows:<br>0 - OK<br>1 - Low |
| signal | Integer | Received signal strength indicator (RSSI). The range is 0-9.<br>0-3 is considered very low,<br>4-5 low, otherwise the strength is high |
| receiver_id | Integer | Node ID of receiving node. |
| receiver_type | Integer | Type of receiving node, present only if receiver is not the controller.<br>1 - Standard Keypad<br>2 - Standard I/O<br>3 - Analysed I/O<br>4 - N/A<br>5 - Wireless (hardware version 1 only). Obsolete.<br>6 - Two reader door controller<br>7 - Multi Area Keypad<br>8 - N/A<br>9 - N/A<br>10 - N/A |
| receiver_wireless | Integer | Indicates if the receiving node is an expander sold as being wireless (that is, type I/O with no inputs and outputs). This attribute can be used for example to show the type as being "Wireless" on a GUI if so desired. This attribute is present only if receiver is not the controller. Values: 0 - no, 1 - yes |

**Example: GET wirelesslog/1**

```
{
    "status": "success",
    "data": {
        "wirelesslog": {
            "sensor": "1",
            "event": [
                {
                    "time": "1390642515",
                    "status": "0",
                    "battery": "0",
                    "signal": "6",
                    "receiver_id": "0"
                },
                {
                    "time": "1390642520",
```

```
                                "status": "2",
                                "battery": "0",
                                "signal": "6",
                                "receiver_id": "0"
                        }
                ]
        }
    }
}
```

# 4 WebSocket API

The SPC Web Gateway provides a WebSocket interface that broadcasts events to "subscribing" clients in real time. To establish a WebSocket connection, the client sends a WebSocket handshake request to the URL:

```
ws://<spc_web_gateway_host>:<port>/ws/spc/
```

Default <port> is 8088 but it is configurable.

## 4.1 WebSocket Event Reports

All messages are reported as SIA events embedded in a JSON frame. Encoding is UTF-8. The format of the message is:

```
{
    "status": "success",
    "data": {
        "sia": {
            "device_id": "<device_id>",
            "timestamp": "<timestamp>",
            "sia_code": "<sia_code>",
            "sia_address": "<sia_address>",
            "description": "<description>",
            "flags": "<flags>",
            "verification_id": "<verification_id>"
        }
    }
}
```

| Element | Type | Description |
|---|---|---|
| device_id | Integer | ID of the device |
| timestamp | Timestamp | Event timestamp |
| sia_code | String | Two character code specifying the event. Please see [SPC_INST_CONF] section 21.7 for valid SIA codes. |
| sia_address | Integer | Number that depend on the SIA code. |
| description | String | Textual description of the event |
| flags | String | Each flag is a single character, and more than one flag can be concatenated to form a string. If there are no flags the flags string will be empty.<br>A – Area Fullset<br>D – Area Unset<br>I – Zone Isolated or Unisolated<br>U – Door Unlocked<br>V – Event has video verification data<br>W – Event has audio verification data<br>1 – Event is being sent by primary modem<br>2 – Event is being sent by backup modem |
| verification_id | Integer | Zero means the panel has no audio or video information for the event, otherwise it is the Verification Zone ID. |

**Example: Burglary Alarm**

LUNDIX IT

```
{
        "status": "success",
        "data": {
                "sia": {
                        "device_id": "1000",
                        "timestamp": "1390645123",
                        "sia_code": "BA",
                        "sia_address": "1",
                        "description": "Front door",
                        "flags": "V",
                        "verification_id": "1"
                }
        }
}
```

Device 1000, timestamp (1390645123) 25 Jan 2014 10:18:43 GMT, Burglary Alarm, Zone 1, Zone Name is Front door, Video is available for this event, Verification Zone ID is 1.

## 4.2 JavaScript WebSocket example

This is a code example written in JavaScript showing how to establish and read messages on a SPC Web Gateway WebSocket connection. The code is based on node.js and websocket.js.

```javascript
var websocket_client = require('websocket').client;

// Connect to SPC Web Gateway websocket interface on 192.168.1.100:8088
var ws_client = new websocket_client();
ws_client.connect('ws://192.168.1.100:8088/ws/spc');

ws_client.on('connectFailed', function(error) {
    console.log('Connect Error: ' + error.toString());
});

ws_client.on('connect', function(connection) {
    console.log('WebSocket client connected');
    connection.on('error', function(error) {
        console.log("Connection Error: " + error.toString());
    });
    connection.on('close', function() {
        console.log('echo-protocol Connection Closed');
    });
    connection.on('message', function(message) {
        if (message.type === 'utf8') {
            manageSiaEvent(message.utf8Data);
        }
    });
});

// manageSiaEvent
function manageSiaEvent(message){
    var msg = JSON.parse(message);
    if (msg.status === 'success'){
        var sia_code = msg.data.sia.sia_code;

        // Take care of SIA event action
```

```
        switch (sia_code){
            case 'BA': /* Burglar Alarm */
                …
                break;
            case 'BR': /* Burglar Alarm Restore */
                …
                break;
            case 'BB': /* Inhibited or Isolated */
                …
                break;
            …
        }
    }
}
```

END OF DOCUMENT