Network Based
Occupancy Sensor

JSON configuration and communication
1.0

AT-OCS-900N

Atlona Manuals
Automation

Version Information

Version	Release Date	Notes
3	Feb 2023	- Corrected syntax for set:led. - Corrected quotation marks in JSON code examples, which were causing errors during JSON validation.

Table of Contents

Introduction

The following is a list of JSON commands for the AT-OCS-900N. Commands are case-sensitive. Do not change spacing or lettering. Each command is terminated with a carriage return. If the command fails or is entered incorrectly, then the feedback is "Invalid command".

⚠ **IMPORTANT:** Each command must be in JSON structure. Each reply from the device, regardless of type, will include an `error` field. If this is set to `true`, then the request failed.

Refer to the following table for port assignments:

Protocol	Port
MQTT	1883
TCP/UDP	9000
WebSocket Secure	443
WebSocket	80

WebSocket Overview

The JSON API offers a structured way to retrieve and manipulate the configuration of AT-OCS-900N devices. The configuration is organized into a number of different configuration nodes, which can be individually retrieved using `get` calls and individually changed using `set` calls. For operations which do not fit neatly into this concept, such as `reboot`, the JSON API also offers the `method` call.

The API uses WebSocket as a transport layer. The WebSocket layer provides full-duplex, reliable message-oriented communication.

Each request consists of a JSON object, sent as a WebSocket message. Each request will get a single message and a JSON object as a reply.

The API makes no use of WebSocket subprotocols.

The AT-OCS-900N makes use of two control URL addresses, depending whether or not Transport Layer Security (TLS) is enabled or disabled:

- If TLS is *enabled*, then the control URL will be `wss://[ip-address-of-AT-OCS-900N]/ws`.
- If TLS is *disabled*, then the control URL will be `ws://[ip-address-of-AT-OCS-900N]/ws`.

Refer to the AT-OCS-900N User Manual for more information on using TLS.

The lifetime of the connection has no meaning within the protocol. It's possible to send multiple requests within a single connection, or to set up a new WebSocket connection for each request. For efficiency, it is recommended to keep the WebSocket connection open and send multiple requests using the same connection, but it is not required.

In addition to WebSocket Secure on port 443 when TLS is enabled (default) and WebSocket on port 80 when TLS is disabled. The AT-OCS-900N supports the same JSON message structure on port 9000 when using TCP or UDP.

Refer to https://www.rfc-editor.org/rfc/rfc6455.txt for the WebSocket specification.

Refer to https://json-schema.org/draft/2020-12/json-schema-core.html for the JSON specification.

Asynchronous Use

NOTE: Asynchronous communication is not supported for methods.

Optionally each request, of any type, may also include an ‘id’ field with an arbitrary string as value. This string is selected by the client, and returned in the ‘id’ field of the reply. This is useful for asynchronous implementations as it allows them to match the reply with the request.

```
Request: {
  "id": "foo",
  "config_set": {
    "config": [{
      "enabled": true,
      "name": "ip_input0"
    }],
    "name": "ip_input"
  },
  "password": "3243F6A8885",
  "username": "admin"
}
Reply: {
  "error": false,
  "id": "foo"
}
```

The device does not parse or alter the value of the ‘id’ field. It can be set to any value. Ensuring its uniqueness across requests is the responsibility of the client.

Request Types

Get

The `get` request is used to retrieve information from the device. It takes only a single parameter: the name of the configuration node to retrieve.

Set

Configuration changes are made through the `set` request.

The `set` request has two important fields: `name` is the identifier of the configuration node to change, and the `config` field contains the configuration as a JSON object or list.

The content of the `config` node shares the same structure as the content of the `config` node in the reply to the `get` request, except that it does not include the read-only fields.

Method

The `method` request is generally used to make changes on the device which may not be persistent. For example, it's used to reboot the device, reset it to its factory defaults or to request the list of available config nodes for a `get` call.

The `method` request includes an object whose name determines the action to be performed. The fields inside the object can be considered the arguments to the `method` request.

The `method` JSON object must contain exactly one sub-object.

Commands

Command	Description
get:deviceInfo	General device information
get:hardwareInfo	Sensor and indicator information
get:info	Return / set device information
get:led	LED indicator information
get:light	Light sensor information
get:motion	Motion sensor information
get:network	Network settings
get:networkInfo	Network information
get:protocolInfo	Protocol information
get:protocols	The communication protocols on the device
get:sensors	Sensor information
get:systemInfo	General system information
get:temperature	Temperature sensor information
get:user	The user account

Command	Description
set:info	Sets device information
set:led	LED indicator information
set:light	Light sensor information
set:motion	Motion sensor information
set:network	Network settings
set:networkInfo	Network information
set:protocolInfo	Protocol information
set:protocols	The communication protocols on the device
set:sensors	Sensor information
set:systemInfo	General system information
set:temperature	Temperature sensor information
set:user	The user account

Command	Description
method:factory_default	Factory reset
method:help	Listing of get, set, and method nodes
method:identify	Light sensor information
method:reboot	Reboots the unit

get:deviceInfo

Returns general device information. For more detailed information about the device, use the `get:info` node.

Syntax
`{"get":"deviceInfo"}`

Example

```
{"get":"deviceInfo"}
```

Feedback

```
{
  "error": false,
  "config": {
    "deviceInfo": {
      "model": "AT-OCS-900N",
      "hardwareRevision": "A4.1",
      "serialNumber": "0710202221102600156",
      "firmwareVersion": "1.0.0",
      "uptime": "0:00:05:47"
    }
  }
}
```

get:hardwareInfo

Returns Sensor and indicator information.

Syntax
`{"get":"hardwareInfo"}`

Example

```
{"get":"hardwareInfo"}
```

Feedback

```
{
  "error": false,
  "config": {
    "hardwareInfo": {
      "sensors": {
        "motion": {
          "detected": true,
          "lastDetection": "0:00:00:26",
          "activeDelay": 3,
          "noDelay": 3
        },
        "light": {
          "values": {
            "white": 3152,
            "red": 845,
            "green": 883,
            "blue": 427
          },
          "sensitivity": 5,
          "events": {
            "unit": "ADC",
            "threshold": 300
          }
        },
        "temperature": {
          "values": {
            "ADC": "Stabilizing",
            "F": "Stabilizing",
            "C": "Stabilizing"
          },
          "events": {
            "unit": "F",
            "threshold": 2.0
          }
        }
      },
      "led": {
        "rgb": "000000",
        "center": true,
        "outer": false,
        "auto": true,
        "motionTrue": "FF0000",
        "motionFalse": "000000"
      }
    }
  }
}
```

get:info

Returns device information.

Syntax
`{"get":"info"}`

Example

```
{"get":"info"}
```

Feedback

```
{
  "error": false,
  "config": {
    "info": {
      "deviceInfo": {
        "model": "AT-OCS-900N",
        "hardwareRevision": "A4.1",
        "serialNumber": "07102022211026001 56",
        "firmwareVersion": "1.0.0",
        "uptime": "0:00:39:35"
      },
      "systemInfo": {
        "name": "Sensor",
        "location": "Not Assigned"
      },
      "networkInfo": {
        "macAddress": "B8-98-B0-0E-45-10",
        "hostname": "OCS-900N-0E4510",
        "tls": true,
        "ipSettings": {
          "dhcp": true,
          "ip": "10.1.0.24",
          "netmask": "255.255.254.0",
          "gateway": "10.1.1.254",
          "dns": {
            "primary": "0.0.0.0",
            "secondary": "0.0.0.0"
          }
        }
      },
      "protocolInfo": {
        "tcp": true,
        "udp": true,
        "ws": true,
        "mqtt": false
      }
    }
  }
} {
  "error": false,
  "config": {
    "info": {
     ...                          // Example feedback truncated
     ...
}}}
```

get:led

Returns LED indicator information.

Syntax
`{"get":"led"}`

Example

```
{"get":"led"}
```

Feedback

```
{
  "error": false,
  "config": {
    "led": {
      "rgb": "000000",
      "center": true,
      "outer": false,
      "auto": true,
      "motionTrue": "FF0000",
      "motionFalse": "000000"
    }
  }
}
```

get:light

Returns LED indicator information.

Syntax
`{"get":"light"}`

Example

```
{"get":"light"}
```

Feedback

```
{
  "error": false,
  "config": {
    "light": {
      "values": {
        "white": 2408,
        "red": 643,
        "green": 685,
        "blue": 333
      },
      "sensitivity": 5,
      "events": {
        "unit": "ADC",
        "threshold": 300
      }
    }
  }
}
```

get:motion

Returns motion sensor information.

Syntax
`{"get":"motion"}`

Example

```
{"get":"motion"}
```

Feedback

```
{
  "error": false,
  "config": {
    "motion": {
      "detected": true,
      "lastDetection": "0:00:00:09",
      "activeDelay": 3,
      "noDelay": 3
    }
  }
}
```

get:network

Returns network settings for the device.

Syntax
`{"get":"network"}`

Example

```
{"get":"network"}
```

Feedback

```
{
  "error": false,
  "config": {
    "network": {
      "macAddress": "B8-98-B0-0E-45-10",
      "hostname": "OCS-900N-0E4510",
      "tls": true,
      "ipSettings": {
        "dhcp": true,
        "ip": "10.1.0.24",
        "netmask": "255.255.254.0",
        "gateway": "10.1.1.254",
        "dns": {
          "primary": "0.0.0.0",
          "secondary": "0.0.0.0"
        }
      }
    }
  }
}
```

get:networkInfo

Returns network information.

Syntax
`{"get":"networkInfo"}`

Example

```
{"get":"networkInfo"}
```

Feedback

```
{
  "error": false,
  "config": {
    "networkInfo": {
      "macAddress": "B8-98-B0-0E-45-10",
      "hostname": "OCS-900N-0E4510",
      "tls": true,
      "ipSettings": {
        "dhcp": true,
        "ip": "10.1.0.24",
        "netmask": "255.255.254.0",
        "gateway": "10.1.1.254",
        "dns": {
          "primary": "0.0.0.0",
          "secondary": "0.0.0.0"
        }
      }
    }
  }
}
```

get:protocolInfo

Returns protocol information.

Syntax
`{"get":"protocolInfo"}`

Example

```
{"get":"protocolInfo"}
```

Feedback

```
{
  "error": false,
  "config": {
    "protocolInfo": {
      "tcp": true,
      "udp": true,
      "ws": true,
      "mqtt": false
    }
  }
}
```

get:protocols

Returns the state of the communication protocols on the device.

Syntax
`{"get":"protocols"}`

Example

```
{"get":"protocols"}
```

Feedback

```
{
  "error": false,
  "config": {
    "protocols": {
      "tcp": {
        "enabled": true,
        "port": 9000
      },
      "ws": {
        "enabled": true,
        "port": 443
      }
    }
  }
} {
  "error": false,
  "config": {
    "protocols": {
      "udp": {
        "enabled": true,
        "port": 9000,
        "subscribers": {
          "1": {
            "enabled": false,
            "ip": "0.0.0.0",
            "port": "",
            "motion": false,
            "light": false,
            "temperature": false
          },
          "2": {
            "enabled": false,
            "ip": "0.0.0.0",
            "port": "",
            "motion": false,
            "light": false,
            "temperature": false
          },
        }
      }
    }
  }
  ...                          // Example feedback truncated
  ...
}
```

get:sensors

Returns sensor information.

Syntax
`{"get":"sensors"}`

Example

```
{"get":"sensors"}
```

Feedback

```
{
  "error": false,
  "config": {
    "sensors": {
      "motion": {
        "detected": true,
        "lastDetection": "0:00:00:22",
        "activeDelay": 3,
        "noDelay": 3
      },
      "light": {
        "values": {
          "white": 2474,
          "red": 665,
          "green": 716,
          "blue": 349
        },
        "sensitivity": 5,
        "events": {
          "unit": "ADC",
          "threshold": 300
        }
      },
      "temperature": {
        "values": {
          "ADC": 935,
          "F": 73.9,
          "C": 23.3
        },
        "events": {
          "unit": "F",
          "threshold": 2.0
        }
      }
    }
  }
}
```

get:systemInfo

Returns general system information.

Syntax
{"get":"systemInfo"}

Example

```
{"get":"systemInfo"}
```

Feedback

```
{
  "error": false,
  "config": {
    "systemInfo": {
      "name": "Sensor",
      "location": "Not Assigned"
    }
  }
}
```

get:temperature

Returns temperature sensor information.

Syntax
{"get":"temperature"}

Example

```
{"get":"temperature"}
```

Feedback

```
{
  "error": false,
  "config": {
    "temperature": {
      "values": {
        "ADC": 937,
        "F": 74.3,
        "C": 23.5
      },
      "events": {
        "unit": "F",
        "threshold": 2.0
      }
    }
  }
}
```

get:user

Returns user account information.

Syntax
`{"get":"user"}`

Example

```
{"get":"user"}
```

Feedback

```
{
  "error": false,
  "config": {
    "user": {
      "username": "admin",
      "passwordHash": "N/A"
    }
  }
}
```

set:info

Sets device information.

Syntax

```
{
  "set": {
    "name": "info",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "info",
    "config": {
      "networkInfo": {
        "hostname": "sensor1",
        "tls": false,
        "ipSettings": {
          "dhcp": true
        }
      },
      "protocolInfo": {
        "tcp": false
      }
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "info": {
      "networkInfo": {
        "hostname": "sensor1",
        "tls": false,
        "ipSettings": {
          "dhcp": true
        }
      },
      "protocolInfo": {
        "tcp": false
      }
    }
  }
}
```

set:led

Sets LED indicator information.

Syntax

```
{
  "set": {
    "name": "led",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "led",
    "config": {
      "outer": true,
      "auto": false,
      "motionTrue": "98f542",
      "motionFalse": "000000"
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "led": {
      "rgb": "000000",
      "center": false,
      "outer": true,
      "auto": false,
      "motionTrue": "98F542",
      "motionFalse": "000000"
    }
  }
}
```

set:light

Sets LED indicator information.

Syntax

```
{
  "set": {
    "name": "light",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "light",
    "config": {
      "sensitivity": 5,
      "events": {
        "threshold": 500
      }
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "light": {
      "sensitivity": 5,
      "events": {
        "threshold": 500
      }
    }
  }
}
```

set:motion

Sets motion sensor information.

Syntax

```
{
  "set": {
    "name": "motion",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "motion",
    "config": {
      "delay": 5
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "motion": {
      "delay": 5
    }
  }
}
```

set:network

Sets network settings for the device.

Syntax

```
{
  "set": {
    "name": "network",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "network",
    "config": {
      "hostname": "sensor1",
      "tls": false,
      "ipSettings": {
        "dhcp": false,
        "ip": "192.168.1.254",
        "netmask": "/24",
        "gateway": "192.168.1.1",
        "dns": {
          "primary": "192.168.1.1",
          "secondary": "10.1.1.1"
        }
      }
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "network": {
      "hostname": "sensor1",
      "tls": false,
      "ipSettings": {
        "dhcp": false,
        "ip": "192.168.1.254",
        "netmask": "/24",
        "gateway": "192.168.1.1",
        "dns": {
          "primary": "192.168.1.1",
          "secondary": "10.1.1.1"
        }
      }
    }
  }
}
```

set:networkInfo

Sets network settings for the device.

Syntax

```
{
  "set": {
    "name": "networkInfo",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "networkInfo",
    "config": {
      "hostname": "sensor1",
      "tls": false,
      "ipSettings": {
        "dhcp": false,
        "ip": "192.168.1.254",
        "netmask": "/24",
        "gateway": "192.168.1.1",
        "dns": {
          "primary": "192.168.1.1",
          "secondary": "10.1.1.1"
        }
      }
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "networkInfo": {
      "hostname": "sensor1",
      "tls": false,
      "ipSettings": {
        "dhcp": false,
        "ip": "192.168.1.254",
        "netmask": "/24",
        "gateway": "192.168.1.1",
        "dns": {
          "primary": "192.168.1.1",
          "secondary": "10.1.1.1"
        }
      }
    }
  }
}
```

set:protocolInfo

Sets protocol information for the device.

Syntax

```
{
  "set": {
    "name": "protocolInfo",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "protocolInfo",
    "config": {
      "tcp": false
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "protocolInfo": {
      "tcp": false
    }
  }
}
```

set:protocols

Sets the communication protocols on the device.

Syntax

```
{
  "set": {
    "name": "protocols",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "protocols",
    "config": {
      "udp": {
        "subscribers": {
          "1": {
            "enabled": true,
            "ip": "192.168.1.50",
            "port": 9000,
            "motion": true,
            "light": true,
            "temperature": true
          }
        }
      }
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "protocols": {
      "udp": {
        "subscribers": {
          "1": {
            "enabled": true,
            "ip": "192.168.1.50",
            "port": 9000,
            "motion": true,
            "light": true,
            "temperature": true
          }
        }
      }
    }
  }
}
```

set:sensors

Sets sensor information.

Syntax

```
{
  "set": {
    "name": "sensors",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "sensors",
    "config": {
      "motion": {
        "delay": 5
      },
      "light": {
        "sensitivity": 5,
        "events": {
          "threshold": 500
        }
      },
      "temperature": {
        "events": {
          "unit": "C",
          "threshold": 5
        }
      }
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "sensors": {
      "motion": {
        "delay": 5
      },
      "light": {
        "sensitivity": 5,
        "events": {
          "threshold": 500
        }
      },
      "temperature": {
        "events": {
          "unit": "C",
          "threshold": 5
        }
      }
    }}}
```

set:systemInfo

Sets general system information.

Syntax

```
{
  "set": {
    "name": "systemInfo",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "systemInfo",
    "config": {
      "name": "sensor1",
      "location": "Executive Conference Room"
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "systemInfo": {
      "name": "sensor1",
      "location": "Executive Conference Room"
    }
  }
}
```

set:temperature

Sets general system information.

Syntax

```
{
  "set": {
    "name": "temperature",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "temperature",
    "config": {
      "events": {
        "unit": "C",
        "threshold": 5
      }
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "temperature": {
      "events": {
        "unit": "C",
        "threshold": 5
      }
    }
  }
}
```

set:user

Sets user account information.

Syntax

```
{
  "set": {
    "name": "user",
    "config": {
      …}
  }
}
```

Example

```
{
  "set": {
    "name": "user",
    "config": {
      "username": "admin",
      "password": "Atlona"
    }
  }
}
```

Feedback

```
{
  "error": false,
  "config": {
    "user": {
      "username": "admin",
      "password": "Atlona"
    }
  }
}
```

method:factory_default

Resets the unit to factory-default settings.

Syntax
`{"method":"factory_default"}`

Example

```
{"method":"factory_default"}
```

Feedback

```
{
  "error": false,
  "reply": "Resetting to Factory Default and Rebooting…"
}
```

method:help

Displays a list of available `get`, `set`, and `method` nodes.

Syntax
`{"method":"help"}`

Example

```
{"method":"help"}
```

Feedback

```
{
  "error": false,
  "reply": {
    "get": [
      "info", "deviceInfo", "systemInfo", "hardwareInfo", "networkInfo",
"protocolInfo", "sensors", "motion", "light", "temperature", "led", "protocols",
"network", "user"
    ],
    "set": [
      "info", "systemInfo", "networkInfo", "protocolInfo", "sensors", "motion",
"light", "temperature", "led", "protocols", "network", "user"
    ],
    "method": [
      "identify", "reboot", "factory_default", "export_config", "import_config",
"upgrade", "help"
    ]
  }
}
```

method:identify

Displays light sensor information.

Syntax
`{"method":"identify"}`

Example

```
{"method":"identify"}
```

Feedback

```
{
  "error": false
} {
  "light": {
    "values": {
      "white": 3523,
      "red": 883,
      "green": 955,
      "blue": 467
    }
  }
} {
  "light": {
    "values": {
      "white": 3830,
      "red": 1129,
      "green": 1236,
      "blue": 649
    }
  }
}
```

method:reboot

Reboots the unit.

Syntax
`{"method":"reboot"}`

Example

```
{"method":"reboot"}
```

Feedback

```
{
  "error": false,
  "reply": "Rebooting…"
}
```