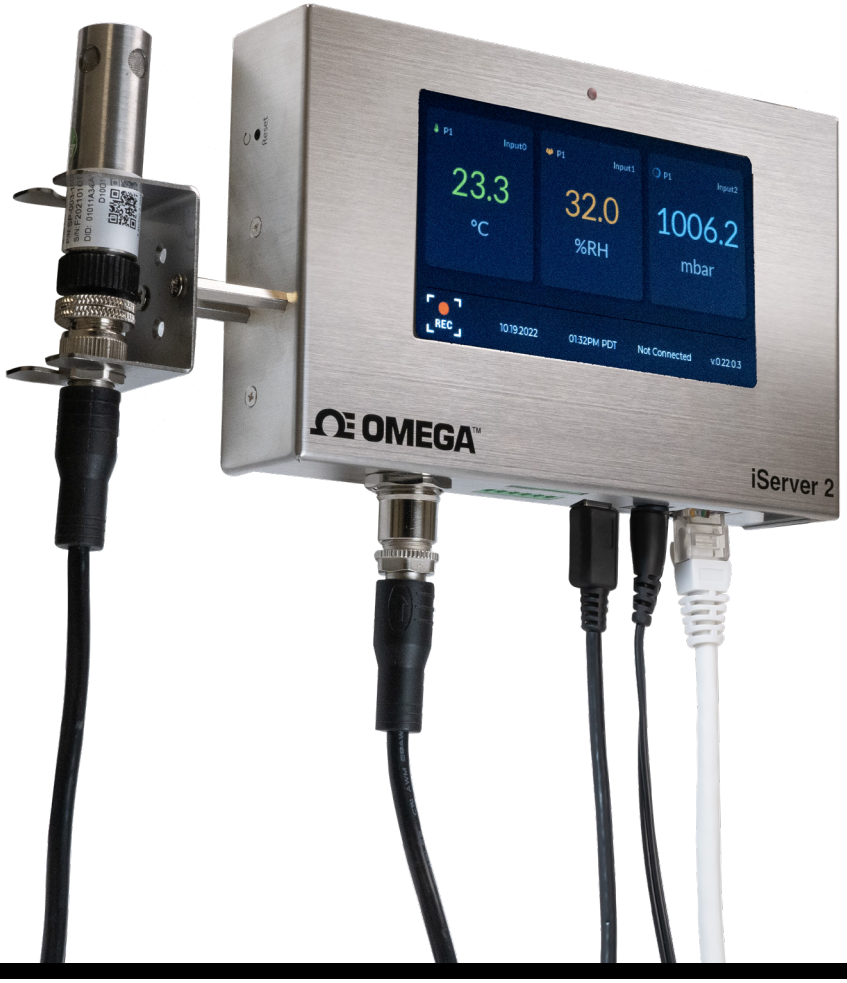
OMEGA™
a DwyerOmega brand

USER'S GUIDE

iServer 2

JSON Websocket Guide



omega.com I info@omega.com

For latest product manuals:
omega.com/en-us/pdf-manuals

1 YEAR WARRANTY

CONTACT

Omega Engineering, Inc.

omega.com/contact-us

Toll-Free:
1-800-826-6342 (USA & Canada only)

Customer Service:
1-800-622-2378 (USA & Canada only)

Engineering Service:
1-800-872-9436 (USA & Canada only)

Telephone:
(203) 359-1660

Fax:
(203) 359-7700

Email: info@omega.com

For other locations visit:
omega.com/worldwide

OMEGA™
a **DwyerOmega** brand

Table of Contents

1. Introduction

This document provides information regarding the JSON objects used in the iServer2.

2. Connecting to the IServer2

Open a WebSocket connection to http://<IServer2 IP address>:8081. Messages with JSON objects can then be used to communicate with the iServer2.

3. JSON Commands

3.1. Login Command

The login command, logs into the iServer2, the return response contains a token that must be sent in other commands.

3.1.1. Command Parameters

```
{
  "login": {
    "username": "string",
    "password": "string"
  }
}
```

Name	Type	Description
username	String	The username is either "admin" or "user".
password	string	The password to login.

3.1.2. Response Content

```
{
  "login": {
    "status": "string",
    "token": "string"
  }
}
```

Name	Type	Description
status	String	Returns success upon successful login, otherwise an authentication error will return.
token	string	Security token. It will be required for other commands.

3.1.3. Example

Command:

```
{
  "login": {
    "username": "admin",
    "password": "00000000"
  }
}
```

Response:

```
{
  "login": {
    "status": "success",
    "token": "cb2fc968-5234-483c-8ded-8be8422abe6f"
  }
}
```

3.2. Probe List Command

This command retrieves the list of probes and their status. Probe id 0 refers to thermocouples and is meaningless in probe models of the iServer2.

3.2.1. Command Parameters

```
{
  "probeList": {
    "token": "string"
  }
}
```

Name	Type	Description
token	String	Security token from successful login.

3.2.2. Response Content

```
{
  "probeList": {
    "status": "string",
    "probes": [
      {
        "probeId": int,
        "probe": int,
        "sensors": [array],
        "connected": int,
        "timestamp": long,
        "isLock": int,
        "isExtractIP": int
      },
      {
        ...
      },
    ]
```

```
  }
}
```

Name	Type	Description
status	String	Returns success upon successful login, otherwise an authentication error will return.
probes	array	Security token. It will be required for other commands.
probeId	int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
probe	int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
sensors	array	Object that contains sensor ids up to the number of sensors the probe has.
connected	int	0-probe is not connected, 1-probe is connected
timestamp	Long	UTC timestamp
isLock	int	0-probe is not locked by password, 1-probe is locked by password
isExtractIP	int	0-iServer2 is not extracting log data from probe, 1-iServer2 is extracting log data from probe

3.2.3. Example

Command:

```
{
  "probeList": {
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```

Response:

```
{
  "probeList": {
    "status": "success",
    "probes": [
      {
        "probeId": 0,
        "probe": 0,
        "sensors": [],
        "connected": 0,
        "timestamp": 0,
        "isLock": 0,
        "isExtractIP": 0
      },
      {
        "probeId": 1,
        "probe": 1,
```

```
      "sensors": [
        0,
        1,
        2
      ],
      "connected": 1,
      "timestamp": 1706144586,
      "isLock": 0,
      "isExtractIP": 0
    },
    {
      "probeId": 2,
      "probe": 2,
      "sensors": [],
      "connected": 0,
      "timestamp": 0,
      "isLock": 0,
      "isExtractIP": 0
    }
  ]
 }
}
```


3.3. Sensor Data Command

This command is used to read the current display data.

3.3.1. Command Parameters


```
{
 "sensorData": {
   "probe": int,
   "channel": int,
   "token": "string"
 }
}
```


Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
token	String	Security token from successful login.

3.3.2. Response Content

```
{
  "sensorData": {
    "probe": int,
    "channel": int,
    "time": long,
    "value": float,
    "precision": int
  }
}
```

Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
Time	Long	UTC timestamp
value	float	Channel or sensor reading
precision	Int	Number of decimal places for the intended resolution

3.3.3. Example

Command:

```
{
  "sensorData": {
    "probe": 1,
    "channel": 1,
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```

Response:

```
{
  "sensorData": {
    "probe": 1,
    "channel": 1,
    "time": 1706231919,
    "value": 52.900001525878906,
    "precision": 1
  }
}
```

3.4. Sensor Meta Command

This command is used to read the units, the type and the name of sensor.

3.4.1. Command Parameters

```
{
  "sensorMeta": {
    "probe": int,
    "channel": int,
    "token": "string"
  }
}
```

Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
token	String	Security token from successful login.

3.4.2. Response Content

```
{
  "sensorMeta": {
    "status": "string",
    "probe": int,
    "channel": int,
    "type": int,
    "typestr": "string",
    "unit": "string",
    "subtype": int,
    "name": "string",
    "precision": int
  }
}
```

Name	Type	Description
status	string	Returns success if data exists otherwise an error will return
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
type	int	Numeric representation of the sensor type, internal use.
typestr	string	The string representation of the type of sensor
unit	String	The unit of the sensor
subtype	Int	Internal use.
name	string	The name of the sensor.
precision	Int	Number of decimal places for the intended resolution

3.4.3. Example

Command:


```
{
  "sensorMeta": {
    "probe": 1,
    "channel": 0,
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "sensorMeta": {
    "status": "success",
    "probe": 1,
    "channel": 0,
    "type": 1,
    "typestr": "temperature",
    "unit": "C",
    "subtype": 0,
    "name": "Temperature",
    "precision": 1
  }
}
```


3.5. Probe Meta Command

This command is used to read information about the probe.

3.5.1. Command Parameters


```
{
  "probeMeta": {
    "probe": int,
    "token": "string"
  }
}
```


Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
token	String	Security token from successful login.

3.5.2. Response Content

```
{
  "probeMeta": {
    "probe": int,
    "sensor": int,
    "output": int,
    "firmwareVer": long,
    "coreVer": long,
    "manufacturedDate": long,
    "calibratedDate": long,
    "status": "string",
    "probeModel": "string"
  }
}
```

Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
sensor	int	Number of sensors on the probe
output	int	Number of outputs on the probe
firmwareVer	Long	Decimal representation of the firmware version, convert to hex to get the version number (e.g. 2.6.4.0 is 0x02060400)
coreVer	long	Decimal representation of the smart sensor core version, covert to hex to get version number (e.g. 2.6.4.0 is 0x02060400).
manufacturedDate	long	Timestamp of the manufactured date. Requires adding 946684800 to the value to get the UTC timestamp.
calibratedDate	long	Timestamp of time of last calibration date. Requires adding 946684800 to the value to get the UTC timestamp.
status	string	Returns success if data exists otherwise an error will return
probeModel	string	The model of the probe attached.

3.5.3. Example

Command:

```
{
  "probeMeta": {
    "probe": 1,
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```

Response:

```
{
  "probeMeta": {
```

```
    "probe": 1,
    "sensor": 3,
    "output": 2,
    "firmwareVer": 33948672,
    "coreVer": 54793475,
    "manufacturedDate": 84871141,
    "calibratedDate": 84871141,
    "status": "success",
    "probeModel": "SP-003-1"
  }
}
```


3.6. System Meta Command

This command is used to read information about the iServer2.

3.6.1. Command Parameters


```
{
  "systemMeta": {
    "token": "string"
  }
}
```


Name	Type	Description
token	String	Security token from successful login.

3.6.2. Response Content


```
{
  "dbupdate": {
    "status": "string"
  },
  "systemMeta": {
    "status": "string",
    "firmwareStr": "string",
    "hardware": long,
    "manufacturer": "string",
    "model": "string",
    "deviceName": "string",
    "systemName": "string",
    "id": "string",
    "samplingTime": int,
```

```
    "probeMode": int
  }
}
```


Name	Type	Description
status	string	Returns success if data exists otherwise an error will return
firmwareStr	string	iServer2 firmware version
hardware	long	reserved
manufacturer	string	Manufacturer of iServer2
model	string	Model of the iServer2
deviceName	string	Name of the iServer2
systemName	long	Name used for DNS discovery
id	string	Serial number of the iServer2
samplingTime	int	Sample time of the iServer2, it is fixed to 1 sample per second
probeMode	int	0 for single probe mode, 1 for dual probe mode

3.6.3. Example

Command:


```
{
  "systemMeta": {
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "dbupdate": {
    "status": "success"
  },
  "systemMeta": {
    "status": "success",
    "firmwareStr": "1.0.7.1",
    "hardware": 16843266,
    "manufacturer": "Omega",
    "model": "iS2-THB-DP",
    "deviceName": "iServer2",
    "systemName": "is2-omegad79d",
    "id": "F2215i030123",
    "samplingTime": 1,
    "probeMode": 0
  }
}
```

3.7. Adjust Info Command (Gain and Offset)

This command is used to read and write a sensor's gain and offset.

3.7.1. Read Command Parameters

```
{
  " adjustInfo": {
    "probe": int,
    "channel": int,
    "token": "string"
  }
}
```

Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
token	String	Security token from successful login.

3.7.2. Read Response Content

```
{
  "adjustInfo": {
    "status": "string",
    "probe": int,
    "channel": int,
    "gain": float,
    "offset": float
  }
}
```

Name	Type	Description
status	string	Returns success if data exists otherwise an error will return
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
gain	float	Gain of the sensor
offset	float	Offset of the sensor

3.7.3. Read Example

Command:

```
{
  " adjustInfo": {
    "probe": 1,
    "channel": 1,
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
```

```
  }
}
```

Response:

```
{
  "adjustInfo": {
    "status": "success",
    "probe": 1,
    "channel": 0,
    "gain": 1.0,
    "offset": 0.0
  }
}
```

3.7.4. Write Command Parameters

The parameters listed as optional do not need to be included in the JSON object.

```
{
  "adjustInfo": {
    "probe": int,
    "channel": int,
    "gain": float, (optional)
    "offset": float, (optional)
    "token": "string"
  }
}
```

Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
gain	float	Gain of the sensor
offset	float	Offset of the sensor
token	String	Security token from successful login

3.7.5. Write Response Parameters

```
{
  "adjustInfo": {
    "status": "string",
    "probe": int,
    "channel": int
  }
}
```

Name	Type	Description
status	string	Returns success if data was successfully written, otherwise an error will return
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3

3.7.6. Write Example

Command:

```
{
  "adjustInfo": {
    "probe": 1,
    "channel": 0,
    "gain": 2.0,
    "offset": 1.0,
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```

Response:

```
{
  "adjustInfo": {
    "status": "success",
    "probe": 1,
    "channel": 0
  }
}
```

3.8. Alarm Info Command

This command is used to read and write a sensor's alarm settings.

3.8.1. Read Command Parameters

```
{
  " alarmInfo": {
    "probe": int,
    "channel": int,
    "token": "string"
  }
}
```

Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
token	String	Security token from successful login.

3.8.2. Read Response Content


```
{
  “alarmInfo”: {
    “status”: “string”,
    “probe”: int,
    “channel”: int,
    “alarmStatus”: int,
    “type”: int,
    “latch”: int,
    “lowThreshold”: float,
    “highThreshold”: float,
    “deadband”: float
  }
}
```


Name	Type	Description
status	string	Returns success if data exists otherwise an error will return
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
alarmStatus	int	0 for no alarm, 1 for alarm low triggered, 2 for alarm high triggered, 3 for alarm high low triggered
type	int	Type of alarm, 0 for no alarm, 1 for alarm low, 2 for alarm high, 3 for alarm high low
latch	int	0 do not latch alarm, 1 latch alarm
lowThreshold	float	Low threshold to trigger on for alarm low and alarm low high
highThreshold	Float	High threshold to trigger on for alarm high and alarm low high
deadband	float	Value the sensor must rise or fall to clear the alarm.

3.8.3. Read Example

Command:


```
{
  “ alarmInfo”: {
    “probe”: 1,
    “channel”: 1,
    “token”: “ cb2fc968-5234-483c-8ded-8be8422abe6f “
  }
}
```


Response:


```
{
  “alarmInfo”: {
```

```
    "status": "success",
    "probe": 1,
    "channel": 0,
    "alarmStatus": 0,
    "type": 0,
    "latch": 0,
    "lowThreshold": 0.0,
    "highThreshold": 0.0,
    "deadband": 5.0
  }
}
```


3.8.4. Write Command Parameters

The parameters listed as optional do not need to be included in the JSON object.


```
{
  "alarmInfo": {
    "probe": int,
    "channel": int,
    "type": int, (optional)
    "lowThreshold": float, (optional)
    "highThreshold": float, (optional)
    "latch": int, (optional)
    "deadband": float, (optional)
    "token": "string"
  }
}
```


Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
type	int	Type of alarm, 0 for no alarm, 1 for alarm low, 2 for alarm high, 3 for alarm high low
latch	int	0 do not latch alarm, 1 latch alarm
lowThreshold	float	Low threshold to trigger on for alarm low and alarm low high
highThreshold	Float	High threshold to trigger on for alarm high and alarm low high
deadband	float	Value the sensor must rise or fall to clear the alarm.
token	String	Security token from successful login.

3.8.5. Write Response Parameters


```
{
  "alarmInfo": {
    "status": "string",
    "probe": int,
    "channel": int
  }
}
```


Name	Type	Description
status	string	Returns success if data was successfully written, otherwise an error will return
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3

3.8.6. Write Example

Command:


```
{
  "alarmInfo": {
    "probe": 1,
    "channel": 1,
    "type": 2,
    "lowThreshold": 0,
    "highThreshold": 60,
    "latch": 0,
    "deadband": 5,
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "alarmInfo": {
    "status": "success",
    "probe": 1,
    "channel": 1
  }
}
```

3.9. Alarm Status Command

This command is used to read a sensor's alarm status.

3.9.1. Command Parameters

```
{
  "alarmStatus": {
    "probe": int,
    "channel": int,
    "token": "string"
  }
}
```

Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
token	String	Security token from successful login.

3.9.2. Response Content

```
{
  "alarmStatus": {
    "probe": int,
    "channel": int,
    "value": int
  }
}
```

Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
value	int	0 for no alarm, 1 for alarm low triggered, 2 for alarm high triggered, 3 for alarm high low triggered

3.9.3. Example

Command:

```
{
  " alarmStatus": {
    "probe": 1,
    "channel": 1,
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```

Response:

```
{
  "alarmStatus": {
    "probe": 1,
    "channel": 1,
    "value": 0
  }
}
```


3.10. Log Config Command

This command is used to read and write the logging interval. This command is also used to enable or disable logging.

3.10.1. Read Command Parameters


```
{
  "logConfig": {
    "token": "string"
  }
}
```


Name	Type	Description
token	String	Security token from successful login

3.10.2. Read Response Content


```
{
  "logConfig": {
    "status": "string",
    "interval": int,
    "enable": int
  }
}
```


Name	Type	Description
status	string	Returns success if data exists otherwise an error will return
Interval	Int	Logging interval in seconds
enable	int	0 disable logging, 1 enable logging

3.10.3. Read Example

Command:


```
{
  " logConfig": {
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```

Response:


```
{
  "logConfig": {
    "status": "success",
    "interval": 5,
    "enable": 1
  }
}
```


3.10.4. Write Command Parameters

The parameters listed as optional do not need to be included in the JSON object.


```
{
  "logConfig": {
    "interval": int, (optional)
    "enable": int, (optional)
    "token": "string"
  }
}
```


Name	Type	Description
interval	Int	Logging interval in seconds
enable	int	0 disable logging, 1 enable logging
token	String	Security token from successful login.

3.10.5. Write Response Parameters


```
{
  "logConfig": {
    "status": "success"
  }
}
```


Name	Type	Description
status	string	Returns success if data was successfully written, otherwise an error will return

3.10.6. Write Example

Command:


```
{
  "logConfig": {
    "interval": 10,
    "enable": 0,
```

```
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```

Response:

```
{
  "logConfig": {
    "status": "success"
  }
}
```

3.11. Probe Config Command

This command is used to read and write the name of the probe.

3.11.1. Read Command Parameters

```
{
  "probeConfig": {
    "probe": int,
    "token": "string"
  }
}
```

Name	Type	Description
probe	int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
token	String	Security token from successful login.

3.11.2. Read Response Content

```
{
  "probeConfig": {
    "status": "string",
    "probe": int,
    "samplingTime": int,
    "deviceName": "string"
  }
}
```

Name	Type	Description
status	string	Returns success if data exists otherwise an error will return
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
samplingTime	int	Fixed at 1 sample per second
deviceName	string	Name of the probe

3.11.3. Read Example

Command:


```
{
  "probeConfig": {
    "probe": 1,
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "probeConfig": {
    "status": "success",
    "probe": 1,
    "samplingTime": 1,
    "deviceName": "Device_06332FD1"
  }
}
```


3.11.4. Write Command Parameters


```
{
  "probeConfig": {
    "probe": int,
    "deviceName": "string",
    "token": "string"
  }
}
```


Name	Type	Description
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
deviceName	string	Probe name, maximum 16 characters
token	String	Security token from successful login.

3.11.5. Write Response Parameters


```
{
  "probeConfig": {
    "status": "string",
    "probe": int
  }
}
```

Name	Type	Description
status	string	Returns success if data was successfully written, otherwise an error will return
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2

3.11.6. Write Example

Command:


```
{
  "probeConfig": {
    "probe": 1,
    "deviceName": "Device_06",
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "probeConfig": {
    "status": "success",
    "probe": 1
  }
}
```


3.12. Alarm (Buzzer) Trigger Command

This command is used to read and write the triggers for the buzzer. The command is also used to enable or disable the buzzer.

3.12.1. Read Command Parameters


```
{
  "alarmTrigger": {
    "apiaction": "string",
    "token": "string"
  }
}
```


Name	Type	Description
apiaction	string	Action either "read" or "write"
token	String	Security token from successful login.

3.12.2. Read Response Content


```
{
  "alarmTrigger": {
    "status": "string",
```

```
    "alarm": {
      "isAlarmEnabled": int,
      "a": reserved,
      "b": int,
      "c": reserved,
      "d": int,
      "e": reserved,
      "f": int,
      "g": int
    },
    "probeEventList": [
      {
        "probeInfo": {
          "probe": int,
          "channel": int,
          "buzzer": int,
          "channelname": "string",
          "probename": "string"
        }
      },
      {
...
        }
      },
    ]
  }
}
```

Name	Type	Description
status	string	Returns success if data exists otherwise an error will return
isAlarmEnabled	int	Enable buzzer, 0 for disable buzzer, 1 for enable buzzer
a	reserved	reserved
b	int	Trigger on any probe disconnecting. 0 for no trigger, 1 for trigger
c	reserved	reserved
d	int	Trigger on out of memory. 0 for no trigger, 1 for trigger
e	reserved	reserved
f	Int	Trigger on low battery. 0 for no trigger, 1 for trigger
g	int	Trigger on TTL contact activation. 0 for no trigger , 1 for trigger
probeEventList	array	Contains the list of probes and sensors and relay trigger information

probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
buzzer	int	Sensor alarm triggers buzzer to turn on, 0 for no effect, 1 for trigger buzzer to turn on
channelname	String	Name of the sensor or channel
probename	string	Name of the probe

3.12.3. Read Example

Command:


```
{
  "alarmTrigger": {
    "apiaction": "read",
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "alarmTrigger": {
    "status": "success",
    "alarm": {
      "isAlarmEnabled": 0,
      "a": 0,
      "b": 0,
      "c": 0,
      "d": 0,
      "e": 0,
      "f": 0,
      "g": 0
    },
    "probeEventList": [
      {
        "probeInfo": {
          "probe": 1,
          "channel": 0,
          "buzzer": 0,
          "channelname": "Temperature",
          "probename": "Device_31312FD1"
```

```
    }
  },
  {
    "probeInfo": {
      "probe": 1,
      "channel": 1,
      "buzzer": 0,
      "channelname": "Humidity",
      "probename": "Device_31312FD1"
    }
  },
  {
    "probeInfo": {
      "probe": 1,
      "channel": 2,
      "buzzer": 0,
      "channelname": "Barometer",
      "probename": "Device_31312FD1"
    }
  },
  {
    "probeInfo": {
      "probe": 2,
      "channel": 0,
      "buzzer": 0,
      "channelname": "Humidex",
      "probename": "Device_10342FD1"
    }
  },
  {
    "probeInfo": {
      "probe": 2,
      "channel": 1,
      "buzzer": 0,
      "channelname": "Dewpoint",
      "probename": "Device_10342FD1"
```

```
        }
      },
      {
        "probeInfo": {
          "probe": 2,
          "channel": 2,
          "buzzer": 0,
          "channelname": "Temperature",
          "probename": "Device_10342FD1"
        }
      },
      {
        "probeInfo": {
          "probe": 2,
          "channel": 3,
          "buzzer": 0,
          "channelname": "Humidity",
          "probename": "Device_10342FD1"
        }
      }
    ]
  }
}
```


3.12.4. Write Command Parameters


```
{
  "alarmTrigger": {
    "apiaction": "string",
    "alarm": {
      "isAlarmEnabled": int,
      "a": reserved,
      "b": int,
      "c": reserved,
      "d": int,
      "e": reserved,
      "f": int,
      "g": int
```

```
    },
    "probeEventList": [
      {
        "probeInfo": {
          "probe": int,
          "channel": int,
          "buzzer": int
        },
      {
...
      }
    ],
    "token": "string"
  }
}
```

Name	Type	Description
apiaction	string	Action either "read" or "write"
isAlarmEnabled	int	Enable buzzer, 0 for disable buzzer, 1 for enable buzzer
a	reserved	reserved
b	int	Trigger on any probe disconnecting. 0 for no trigger, 1 for trigger
c	reserved	reserved
d	int	Trigger on out of memory. 0 for no trigger, 1 for trigger
e	reserved	reserved
f	Int	Trigger on low battery. 0 for no trigger, 1 for trigger
g	int	Trigger on TTL contact activation. 0 for no trigger , 1 for trigger
probeEventList	array	Contains the list of probes and sensors and relay trigger information
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
buzzer	int	Sensor alarm triggers buzzer to turn on, 0 for no effect, 1 for trigger buzzer to turn on
token	String	Security token from successful login.

3.12.5. Write Response Parameters


```
{
  “dbupdate”: {
    “status”: “success”
  }
}
```


Name	Type	Description
status	string	Returns success if data was successfully written, otherwise an error will return

3.12.6. Write Example

Command:


```
{
  “alarmTrigger”: {
    “apiaction”: “write”,
    “alarm”: {
      “isAlarmEnabled”: 0,
      “a”: 0,
      “b”: 0,
      “c”: 0,
      “d”: 0,
      “e”: 0,
      “f”: 1,
      “g”: 0
    },
    “probeEventList”: [
      {
        “probeInfo”: {
          “probe”: 1,
          “channel”: 0,
          “buzzer”: 1
        }
      },
      {
        “probeInfo”: {
          “probe”: 2,
          “channel”: 0,
```

```
        "buzzer": 1
      }
    }
  ],
  "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
 }
} Response:
{
 "dbupdate": {
  "status": "success"
 }
}
```


3.13. Relay Trigger Command

This command is used to read and write the triggers for the relays. The command is also used to check the state of the relay.

3.13.1. Read Command Parameters


```
{
 "relayTrigger": {
  "apiaction": "string",
  "relayId": int,
  "token": "string"
 }
}
```


Name	Type	Description
apiaction	string	Action either "read" or "write"
relayId	int	Relay ID to read, 1 for relay1 and 2 for relay2
token	String	Security token from successful login.

3.13.2. Read Response Content


```
{
 "relayTrigger": {
  "status": "string",
  "relayInfo": {
   "relayId": int,
   "setting": {
    "istate": int,
    "name": "string",
    "isLog": int,
```

```
        "isDisplay": int,
        "isLatch": int,
        "relayStatus": int,
        "a": reserved,
        "b": int,
        "c": reserved,
        "d": int,
        "e": reserved,
        "f": int,
        "g": int
      }
    },
    "probeEventList": [
      {
        "probeInfo": {
          "probe": int,
          "channel": int,
          "relay": int
        }
      },
      {
...
        }
      },
    ]
  }
}
```

Name	Type	Description
status	string	Returns success if data exists otherwise an error will return
relayId	int	Relay identifier, 1 for relay1 and 2 for relay2
istate	int	Initial state of relay, 0 for normally closed, 1 for normally open
name	string	Name of relay
isLog	int	Log relay state changes. 0 for no log, 1 for log
isDisplay	int	Display relay state on web GUI. 0 for no display, 1 for display
isLatch	int	Enable latching. 0 for disable latching, 1 for enable latching
relayStatus	int	State of relay, 0 for closed, 1 for open
a	reserved	reserved
b	int	Trigger on any probe disconnecting. 0 for no trigger, 1 for trigger

c	reserved	reserved
d	int	Trigger on out of memory. 0 for no trigger, 1 for trigger
e	reserved	reserved
f	Int	Trigger on low battery. 0 for no trigger, 1 for trigger
g	int	Trigger on TTL contact activation. 0 for no trigger , 1 for trigger
probeEventList	array	Contains the list of probes and sensors and relay trigger information
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
relay	int	Sensor alarm triggers a relay action for the current relayId, 0 for no effect, 1 for trigger relay action

3.13.3. Read Example

Command:


```
{
  "relayTrigger": {
    "apiaction": "read",
    "relayId": 1,
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "relayTrigger": {
    "status": "success",
    "relayInfo": {
      "relayId": 1,
      "setting": {
        "istate": 1,
        "name": "Relay1",
        "isLog": 0,
        "isDisplay": 1,
        "isLatch": 0,
        "relayStatus": 0,
        "a": 0,
        "b": 0,
        "c": 0,
        "d": 0,
        "e": 0,
        "f": 0,
```

```
      "g": 0
    }
  },
  "probeEventList": [
    {
      "probeInfo": {
        "probe": 1,
        "channel": 0,
        "relay": 1
      }
    },
    {
      "probeInfo": {
        "probe": 1,
        "channel": 1,
        "relay": 1
      }
    },
    {
      "probeInfo": {
        "probe": 1,
        "channel": 2,
        "relay": 1
      }
    },
    {
      "probeInfo": {
        "probe": 2,
        "channel": 0,
        "relay": 0
      }
    },
    {
      "probeInfo": {
        "probe": 2,
        "channel": 1,
```

```
        "relay": 0
      }
    },
    {
      "probeInfo": {
        "probe": 2,
        "channel": 2,
        "relay": 0
      }
    },
    {
      "probeInfo": {
        "probe": 2,
        "channel": 3,
        "relay": 0
      }
    }
  ]
 }
}
```


3.13.4. Write Command Parameters


```
{
  "relayTrigger": {
    "apiaction": "string",
    "relayInfo": {
      "relayId": int,
      "setting": {
        "istate": int,
        "name": "string",
        "isLog": int,
        "isDisplay": int,
        "isLatch": int,
        "a": reserved,
        "b": int,
        "c": reserved,
        "d": int,
```

```
        "e": reserved,
        "f": int,
        "g": int
      }
    },
    "probeEventList": [
      {
        "probeInfo": {
          "probe": int,
          "channel": int,
          "relay": int
        },
      {
...
        }
      }
    ],
    "token": " string "
  }
}
```

Name	Type	Description
apiaction	string	Action either "read" or "write"
relayId	int	Relay identifier, 1 for relay1 and 2 for relay2
istate	int	Initial state of relay, 0 for normally closed, 1 for normally open
name	string	Name of relay
isLog	int	Log relay state changes. 0 for no log, 1 for log
isDisplay	int	Display relay state on web GUI. 0 for no display, 1 for display
isLatch	int	Enable latching. 0 for disable latching, 1 for enable latching
a	reserved	reserved
b	int	Trigger on any probe disconnecting. 0 for no trigger, 1 for trigger
c	reserved	reserved
d	int	Trigger on out of memory. 0 for no trigger, 1 for trigger
e	reserved	reserved
f	Int	Trigger on low battery. 0 for no trigger, 1 for trigger
g	int	Trigger on TTL contact activation. 0 for no trigger , 1 for trigger
probeEventList	array	Contains the list of probes and sensors and relay trigger information
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3

relay	int	Sensor alarm triggers a relay action for the current relayId, 0 for no effect, 1 for trigger relay action
token	String	Security token from successful login.

3.13.5. Write Response Parameters


```
{
  "dbupdate": {
    "status": "success"
  }
}
```


Name	Type	Description
status	string	Returns success if data was successfully written, otherwise an error will return

3.13.6. Write Example

Command:


```
{
  "relayTrigger": {
    "apiaction": "write",
    "relayInfo": {
      "relayId": 1,
      "setting": {
        "istate": 1,
        "name": "relay1",
        "isLog": 0,
        "isDisplay": 1,
        "isLatch": 0,
        "a": 0,
        "b": 0,
        "c": 0,
        "d": 0,
        "e": 0,
        "f": 1,
        "g": 0
      }
    },
    "probeEventList": [
      {
```

```
      "probeInfo": {
        "probe": 1,
        "channel": 0,
        "relay": 1
      }
    },
    {
      "probeInfo": {
        "probe": 2,
        "channel": 0,
        "relay": 1
      }
    }
  ],
  "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
 }
}
```


Response:


```
{
 "dbupdate": {
  "status": "success"
 }
}
```


3.14. Notification (e-mail) Trigger Command

This command is used to read and write the triggers for the relays. The command is also used to check the state of the relay.

3.14.1. Read Command Parameters


```
{
 " notiTrigger": {
  "apiaction": "string",
  "token": "string"
 }
}
```


Name	Type	Description
apiaction	string	Action either "read" or "write"
token	String	Security token from successful login.

3.14.2. Read Response Content

```
{
  "notiTrigger": {
    "status": "string",
    "notiInfo": {
      "isEmailEnabled": int,
      "emailList": "string",
      "heartbeatInterval": int,
      "nextAlarmInterval": int
    },
    "trigger": {
      "a": int,
      "b": int,
      "c": reserved,
      "d": int,
      "e": reserved,
      "f": int,
      "g": int
    },
    "probeEventList": [
      {
        "probeInfo": {
          "probe": int,
          "channel": int,
          "isSend": int
        }
      },
      {
...
      }
    ]
  }
}
```

Name	Type	Description
status	string	Returns success if data exists otherwise an error will return
isEmailEnabled	int	Returns 1
emailList	int	List of e-mails to send the notification to. Semicolons are used to separate multiple e-mail addresses
heartbeatInterval	int	Send a heartbeat notification at a periodic interval. The interval is in minutes.
nextAlarmInterval	int	Resend an alarm notification at a periodic interval. The interval is in minutes.
a	int	Trigger on unit power up.
b	int	Trigger on any probe disconnecting. 0 for no trigger, 1 for trigger
c	reserved	reserved
d	int	Trigger on out of memory. 0 for no trigger, 1 for trigger
e	reserved	reserved
f	Int	Trigger on low battery. 0 for no trigger, 1 for trigger
g	int	Trigger on TTL contact activation. 0 for no trigger, 1 for trigger
probeEventList	array	Contains the list of probes and sensors and relay trigger information
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
isSend	int	Sensor alarm triggers an e-mail notification, 0 for do not send, 1 for send

3.14.3. Read Example

Command:

```
{
  "notiTrigger": {
    "apiaction": "read",
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```

Response:

```
{
  "notiTrigger": {
    "status": "success",
    "notiInfo": {
      "isEmailEnabled": 1,
      "emailList": "test@test.com",
      "heartbeatInterval": 0,
      "nextAlarmInterval": 0
    },
    "trigger": {
      "a": 0,
```

```
    "b": 0,
    "c": 0,
    "d": 0,
    "e": 0,
    "f": 1,
    "g": 0
  },
  "probeEventList": [
    {
      "probeInfo": {
        "probe": 1,
        "channel": 0,
        "isSend": 0
      }
    },
    {
      "probeInfo": {
        "probe": 1,
        "channel": 1,
        "isSend": 0
      }
    },
    {
      "probeInfo": {
        "probe": 1,
        "channel": 2,
        "isSend": 0
      }
    },
    {
      "probeInfo": {
        "probe": 2,
        "channel": 0,
        "isSend": 0
      }
    },
```

```
      {
        "probeInfo": {
          "probe": 2,
          "channel": 1,
          "isSend": 0
        }
      },
      {
        "probeInfo": {
          "probe": 2,
          "channel": 2,
          "isSend": 0
        }
      },
      {
        "probeInfo": {
          "probe": 2,
          "channel": 3,
          "isSend": 0
        }
      }
    ]
  }
}
```

3.14.4. Write Command Parameters

```
{
  "notiTrigger": {
    "apiaction": "string",
    "notiInfo": {
      "emailList": "string",
      "heartbeatInterval": int,
      "nextAlarmInterval": int
    },
    "trigger": {
      "a": int,
      "b": int,
```

```
      "c": reserved,
      "d": int,
      "e": reserved,
      "f": int,
      "g": int
    },
    "probeEventList": [
      {
        "probeInfo": {
          "probe": int,
          "channel": int,
          "isSend": int
        }
      },
      {
...
      }
    ],
    "token": "string"
  }
}
```

Name	Type	Description
apiaction	string	Action either "read" or "write"
emailList	int	List of e-mails to send the notification to. Semicolons are used to separate multiple e-mail addresses
heartbeatInterval	int	Send a heartbeat notification at a periodic interval. The interval is in minutes.
nextAlarmInterval	int	Resend an alarm notification at a periodic interval. The interval is in minutes.
a	int	Trigger on unit power up.
b	int	Trigger on any probe disconnecting. 0 for no trigger, 1 for trigger
c	reserved	reserved
d	int	Trigger on out of memory. 0 for no trigger, 1 for trigger
e	reserved	reserved
f	Int	Trigger on low battery. 0 for no trigger, 1 for trigger
g	int	Trigger on TTL contact activation. 0 for no trigger, 1 for trigger
probeEventList	array	Contains the list of probes and sensors and relay trigger information
probe	Int	Probe number. 0 for thermocouples, 1 for probe 1, 2 for probe 2
channel	int	Channel or sensor number, ranges from 0-3
isSend	int	Sensor alarm triggers an e-mail notification, 0 for do not send, 1 for send

token	String	Security token from successful login.

3.14.5. Write Response Parameters


```
{
  "dbupdate": {
    "status": "success"
  }
}
```


Name	Type	Description
status	string	Returns success if data was successfully written, otherwise an error will return

3.14.6. Write Example

Command:


```
{
  "notiTrigger": {
    "apiaction": "write",
    "notiInfo": {
      "emailList": "test@test.com",
      "heartbeatInterval": 30,
      "nextAlarmInterval": 30
    },
    "trigger": {
      "a": 0,
      "b": 0,
      "c": 0,
      "d": 0,
      "e": 0,
      "f": 1,
      "g": 0
    },
    "probeEventList": [
      {
        "probeInfo": {
          "probe": 1,
          "channel": 0,
          "isSend": 1
```

```
        }
      },
      {
        "probeInfo": {
          "probe": 2,
          "channel": 0,
          "isSend": 1
        }
      }
    ],
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "dbupdate": {
    "status": "success"
  }
}
```


3.15. Contact Closure Config Command

This command is used to configure the digital I/O. The command is also used to check the status of the inputs and the output.

3.15.1. Read Command Parameters


```
{
  " contactClosure": {
    "apiaction": "string",
    "token": "string"
  }
}
```


Name	Type	Description
apiaction	string	Action either "read" or "write"
token	String	Security token from successful login.

3.15.2. Read Response Content


```
{
  "contactClosure": {
    "status": "string",
```

```
    "contact1": {
      "isContactEnabled": int,
      "initiateState": int,
      "contactName": "string",
      "isDisplay": int,
      "isLog": int,
      "OpenStatus": int
    },
    "contact2": {
      "isContactEnabled": int,
      "initiateState": int,
      "contactName": "string",
      "isDisplay": int,
      "isLog": int,
      "OpenStatus": int
    },
    "contactOutput": {
      "isDisplay": int,
      "isLog": int,
      "outputType": int,
      "relayStatus": int,
      "isLatch": int
    }
  }
}
```

Name	Type	Description
status	string	Returns success if data exists otherwise an error will return
isContactEnabled	int	Enable or disable the contact, 0 for disable 1 for enable
initiateState	int	Initial state of contact, 0 for normally closed, 1 for normally open
contactName	string	Name of the contact
isDisplay	int	Display contact state on web GUI. 0 for no display, 1 for display
isLog	int	Log contact state changes. 0 for no log, 1 for log
OpenStatus	int	Contact status. 0 for initial state, 1 for changed state
outputType	int	Output changes based on input. 1 for Active low, 2 for Active high
relayStatus	int	State of output, 0 for closed, 1 for open
isLatch	int	Enable latching. 0 for disable latching, 1 for enable latching

3.15.3. Read Example

Command:


```
{
  " contactClosure": {
    "apiaction": "read",
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "contactClosure": {
    "status": "success",
    "contact1": {
      "isContactEnabled": 1,
      "initiateState": 0,
      "contactName": "Digital Input 1",
      "isDisplay": 1,
      "isLog": 0,
      "OpenStatus": 0
    },
    "contact2": {
      "isContactEnabled": 1,
      "initiateState": 0,
      "contactName": "Digital Input 2",
      "isDisplay": 1,
      "isLog": 0,
      "OpenStatus": 0
    },
    "contactOutput": {
      "isDisplay": 1,
      "isLog": 0,
      "outputType": 2,
      "relayStatus": 1,
      "isLatch": 0
    }
  }
}
```

3.15.4. Write Command Parameters

```
{
  "contactClosure": {
    "apiaction": "string",
    "contact1": {
      "isContactEnabled": int,
      "initiateState": int,
      "contactName": "string",
      "isDisplay": int,
      "isLog": int
    },
    "contact2": {
      "isContactEnabled": int,
      "initiateState": int,
      "contactName": "string",
      "isDisplay": int,
      "isLog": int
    },
    "contactOutput": {
      "isDisplay": int,
      "isLog": int,
      "outputType": int,
      "isLatch": int
    },
    "token": "string"
  }
}
```

Name	Type	Description
apiaction	string	Action either "read" or "write"
isContactEnabled	int	Enable or disable the contact, 0 for disable 1 for enable
initiateState	int	Initial state of contact, 0 for normally closed, 1 for normally open
contactName	string	Name of the contact
isDisplay	int	Display contact state on web GUI. 0 for no display, 1 for display
isLog	int	Log contact state changes. 0 for no log, 1 for log
outputType	int	Output changes based on input. 1 for Active low, 2 for Active high
relayStatus	int	State of output, 0 for closed, 1 for open
isLatch	int	Enable latching. 0 for disable latching, 1 for enable latching
token	String	Security token from successful login.

3.15.5. Write Response Parameters


```
{
  "dbupdate": {
    "status": "success"
  }
}
```


Name	Type	Description
status	string	Returns success if data was successfully written, otherwise an error will return

3.15.6. Write Example

Command:


```
{
  "contactClosure": {
    "apiaction": "write",
    "contact1": {
      "isContactEnabled": 1,
      "initiateState": 0,
      "contactName": "Digital1",
      "isDisplay": 1,
      "isLog": 0
    },
    "contact2": {
      "isContactEnabled": 1,
      "initiateState": 0,
      "contactName": "Digital2",
      "isDisplay": 1,
      "isLog": 0
    },
    "contactOutput": {
      "isDisplay": 1,
      "isLog": 0,
      "outputType": 2,
      "isLatch": 0
    },
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
```

```
}
```

Response:

```
{
  "dbupdate": {
    "status": "success"
  }
}
```

3.16. Relay Clear Latch Command

This command is used to clear the latch on a relay.

3.16.1. Read Command Parameters

```
{
  "relayInfo": {
    "relayId": int,
    "setting": {
      "clearLatch": int
    },
    "apiaction": "string",
    "token": "string"
  }
}
```

Name	Type	Description
relayId	int	Relay ID to clear latch on, 1 for relay1 and 2 for relay2
clearLatch	int	Clear latch on a relay, 1 to clear
apiaction	string	Action either "read" or "write"
token	String	Security token from successful login.

3.16.2. Read Response Content

```
{
  "dbupdate": {
    "status": "string"
  }
}
```

Name	Type	Description
Status	string	Returns success if data exists otherwise an error will return

3.16.3. Example

Command:


```
{
  "relayInfo": {
    "relayId": 1,
    "setting": {
      "clearLatch": 1
    },
    "apiaction": "write",
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "dbupdate": {
    "status": "success"
  }
}
```


3.17. Contact Output Clear Latch Command

This command is used to clear the latch on the output of the contact closures.

3.17.1. Read Command Parameters


```
{
  "contactOutput": {
    "apiaction": "string",
    "clearLatch": int,
    "token": "string"
  }
}
```


Name	Type	Description
apiaction	string	Action either "read" or "write"
clearLatch	int	Clear latch on a relay, 1 to clear
token	String	Security token from successful login.

3.17.2. Read Response Content


```
{
  "dbupdate": {
    "status": "string"
  }
}
```


Name	Type	Description
status	string	Returns success if data exists otherwise an error will return

3.17.3. Example

Command:


```
{
  "contactOutput": {
    "apiaction": "write",
    "clearLatch": 1,
    "token": " cb2fc968-5234-483c-8ded-8be8422abe6f "
  }
}
```


Response:


```
{
  "dbupdate": {
    "status": "success"
  }
}
```


4. CGI Commands

4.1. Extract Data

This command extracts the stored data from the device.

4.1.1. Command Parameters

http://<ip address of iserver2>/cgi-bin/query?startTime=<UTC time>&endTime=<UTC time>&p0ch0=<desired header name>&p0ch1=<desired header name>&p1ch0=<desired header name>&p1ch1=<desired header name>&p1ch2=<desired header name>&p1ch3=<desired header name>&p2ch0=<desired header name>&p2ch1=<desired header name>&p2ch2=<desired header name>&p2ch3=<desired header name>&relay1=<desired header name>&relay2=<desired header name>&contact1=<desired header name>&contact2=<desired header name>&contactoutput=<desired header name>

For TC inputs use p0ch0 and p0ch1 and other fields can be omitted.

For probe inputs use p1ch0-p1ch3 and p2ch0-p2ch3.

Fields relay1, relay2, contact1, contact2, contactout, can be omitted from the command if they are not required.

4.1.2. Example

http://192.168.1.118/cgi-bin/query?startTime=1735847516&endTime=1736452316&p1ch0=Device_31312FD1-Temperature&p1ch1=Device_31312FD1-Humidity&p1ch2=Device_31312FD1-Barometer&p2-ch0=Device_10342FD1-Humidex&p2ch1=Device_10342FD1-Dewpoint&p2ch2=Device_10342FD1-Temperature&p2ch3=Device_10342FD1-Humidity&relay1=Relay1&relay2=Relay2&contact1=Digital%20Input%201&contact2=Digital%20Input%202&contactout=Contact_Out

WARRANTY/DISCLAIMER

OMEGA ENGINEERING, INC. warrants this unit to be free of defects in materials and workmanship for a period of **13 months** from date of purchase. OMEGA's WARRANTY adds an additional one (1) month grace period to the normal **one (1) year product warranty** to cover handling and shipping time. This ensures that OMEGA's customers receive maximum coverage on each product.

If the unit malfunctions, it must be returned to the factory for evaluation. OMEGA's Customer Service Department will issue an Authorized Return (AR) number immediately upon phone or written request. Upon examination by OMEGA, if the unit is found to be defective, it will be repaired or replaced at no charge. OMEGA's WARRANTY does not apply to defects resulting from any action of the purchaser, including but not limited to mishandling, improper interfacing, operation outside of design limits, improper repair, or unauthorized modification. This WARRANTY is VOID if the unit shows evidence of having been tampered with or shows evidence of having been damaged as a result of excessive corrosion; or current, heat, moisture or vibration; improper specification; misapplication; misuse or other operating conditions outside of OMEGA's control. Components in which wear is not warranted, include but are not limited to contact points, fuses, and triacs.

OMEGA is pleased to offer suggestions on the use of its various products. However, OMEGA neither assumes responsibility for any omissions or errors nor assumes liability for any damages that result from the use of its products in accordance with information provided by OMEGA, either verbal or written. OMEGA warrants only that the parts manufactured by the company will be as specified and free of defects. OMEGA MAKES NO OTHER WARRANTIES OR REPRESENTATIONS OF ANY KIND WHATSOEVER, EXPRESSED OR IMPLIED, EXCEPT THAT OF TITLE, AND ALL IMPLIED WARRANTIES INCLUDING ANY WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE HEREBY DISCLAIMED. LIMITATION OF LIABILITY: The remedies of purchaser set forth herein are exclusive, and the total liability of OMEGA with respect to this order, whether based on contract, warranty, negligence, indemnification, strict liability or otherwise, shall not exceed the purchase price of the component upon which liability is based. In no event shall OMEGA be liable for consequential, incidental or special damages.

CONDITIONS: Equipment sold by OMEGA is not intended to be used, nor shall it be used: (1) as a "Basic Component" under 10 CFR 21 (NRC), used in or with any nuclear installation or activity; or (2) in medical applications or used on humans. Should any Product(s) be used in or with any nuclear installation or activity, medical application, used on humans, or misused in any way, OMEGA assumes no responsibility as set forth in our basic WARRANTY/DISCLAIMER language, and, additionally, purchaser will indemnify OMEGA and hold OMEGA harmless from any liability or damage whatsoever arising out of the use of the Product(s) in such a manner.

RETURN REQUESTS/INQUIRIES

Direct all warranty and repair requests/inquiries to the OMEGA Customer Service Department. BEFORE RETURNING ANY PRODUCT(S) TO OMEGA, PURCHASER MUST OBTAIN AN AUTHORIZED RETURN (AR) NUMBER FROM OMEGA'S CUSTOMER SERVICE DEPARTMENT (IN ORDER TO AVOID PROCESSING DELAYS). The assigned AR number should then be marked on the outside of the return package and on any correspondence.

The purchaser is responsible for shipping charges, freight, insurance and proper packaging to prevent breakage in transit.

FOR **WARRANTY** RETURNS, please have the following information available BEFORE contacting OMEGA:

1. Purchase Order number under which the product was PURCHASED,
2. Model and serial number of the product under warranty, and
3. Repair instructions and/or specific problems relative to the product.

FOR **NON-WARRANTY** REPAIRS, consult OMEGA for current repair charges. Have the following information available BEFORE contacting OMEGA:

1. Purchase Order number to cover the COST of the repair,
2. Model and serial number of the product, and
3. Repair instructions and/or specific problems relative to the product.

OMEGA's policy is to make running changes, not model changes, whenever an improvement is possible. This affords our customers the latest in technology and engineering.

OMEGA™

a DwyerOmega brand

omega.com

M-5807/0225

DWYEROMEGA™

Improving the world, one measurement at a time.™

Dwyer OMEGA™ ACT MILJOCO WEISS INSTRUMENTS UFM UNIVERSAL FLOW MONITORS

Smart Probe

Thermocouple