Water Detection Sensor Profile Definition for Rest API

Monnit Corporation

Version 0.1
## Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>8/20/15</td>
<td>created by Brandon Young</td>
</tr>
</tbody>
</table>


Contents
Overview .......................................................................................................................... 4
  General Trigger Sensor Configurations .......................................................................... 4
  General Measurement Sensor Configurations ................................................................. 6
Water Detection Sensor ...................................................................................................... 11
  Sensor Set Calibration Endpoint .................................................................................. 11
  Permitted Values/Ranges .............................................................................................. 12
  Default Values ............................................................................................................. 12
Overview

This document outlines the profile specific values that can be configured using the iMonnit Rest API. The iMonnit database holds the authoritative copy of values for sensor configurations. Storing the values in your application is a common practice. However, certain actions the sensor takes can cause the values to be modified outside API/UI interaction. For example, calibration of a sensor may cause some of the values to be updated after calibration calculations are completed. For this reason you should always retrieve fresh values prior to setting them.

Each sensor profile has a unique numeric application identifier associated with it. Certain endpoints in the REST API this ApplicationID is presented. The list of application identifiers can be found from calling the “GetApplicationID” endpoint.

Sensor profiles are typically categorized into two main categories, measurement sensors and trigger sensors. In this overview we will outline typical values associated with each type. Individual sensor profiles will define custom uses for each of the values but understanding the base use will better enable you to implement multiple profiles with greater ease. Depending on which type of sensor you are configuring will determine which of the available fields are sent to the sensor. Certain fields are only sent to trigger sensors, other fields are only sent to measurement sensors.

A general understanding of Monnit sensor’s Standard and Aware States will be useful for developers to understand while implementing configuration changes in the iMonnit Rest API. Most sensor profiles have two states in which they can report. The Standard State in general provides the sensor with the longest available battery life while performing its required activity. The Aware State allows the sensor to utilize more power if needed when certain conditions are met. These conditions typically will coincide with the application the sensor is being used to monitor. For example: while monitoring the temperature inside a freezer you may be allowed to have longer durations between sensor readings while the temperature is below freezing. This would be the standard state of the sensor. If the temperature rises above freezing (for instance, during a defrost cycle) it may be important to have more frequent data readings to be able to adequately monitor that the freezer is functioning properly or if the freezer is not functioning and the temperature will continue to rise outside permissible operating parameters. Configuring the profile specific values allows you control over how (and when) the Aware State is triggered.

General Trigger Sensor Configurations

Sensor profiles that belong to the Trigger category generally have fewer required values and are easier to configure. Trigger sensors do not utilize the threshold portion of the sensor configurations so the API call to configure those values will not be covered in this section. In the “SensorSetCalibration” endpoint, there are a number of parameters that you must send. Even though some of these values are not utilized by the trigger category all fields are required to be populated. Best practice is to call the “SensorGetCalibration” endpoint and start by populating all the fields with their existing value, then modifying the fields you wish to change. In the following list of parameters the starred (*) fields are fields that are forwarded to the Trigger category of sensors.
**Sensor Set Calibration Parameter List:**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Type</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensorID:</td>
<td>Integer</td>
<td>Unique identifier of the sensor</td>
</tr>
<tr>
<td>calibration1:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>calibration2:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>calibration3:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>calibration4:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>*eventDetectionType:</td>
<td>Integer</td>
<td>Type of event detected</td>
</tr>
<tr>
<td>*eventDetectionCount:</td>
<td>Integer</td>
<td>Number of events required to trigger</td>
</tr>
<tr>
<td>*eventDetectionPeriod:</td>
<td>Integer</td>
<td>Time window for event count to be reached</td>
</tr>
<tr>
<td>*rearmTime:</td>
<td>Integer</td>
<td>Time before event can be triggered again</td>
</tr>
<tr>
<td>*biStable:</td>
<td>Integer</td>
<td>Direction of event</td>
</tr>
<tr>
<td>*pushProfileConfig1:</td>
<td>boolean</td>
<td>Set the configuration page to be pushed to the sensor</td>
</tr>
<tr>
<td>pushProfileConfig2:</td>
<td>boolean</td>
<td>Set the configuration page to be pushed to the sensor</td>
</tr>
<tr>
<td>pushAutoCalibrateCommand:</td>
<td>boolean</td>
<td>Set the auto calibrate command to be pushed to the sensor</td>
</tr>
</tbody>
</table>

**Event Detection Type:** determines which value the sensor will report as its Aware State. Sensors prior to version 2.3.0.0 only allow two values (True/False). Starting with version 2.3.0.0, trigger sensors now support three options (True/False/Change). Change means that if the reading is different than the previous reading it will be marked Aware.

**Event Detection Count:** this field is utilized with the Event Detection Period parameter to create a software filter to adjust sensitivity and prevent false triggers. This is the number of readings the processor must reach within the detection period before it is qualified as an actual trigger event. Each profile adjusts this count differently (check profile specific documentation before modifying this value).

**Event Detection Period:** this field is utilized with the Event Detection Count parameter to create a software filter to adjust sensitivity and prevent false triggers. This is the time window the processor uses to observe readings before it is qualified as an actual trigger event. Each profile calculates this period differently (check profile specific documentation before modifying this value).

**Rearm Time:** once an event is qualified and triggered, this is the amount of time in seconds the sensor is deactivated to prevent unintended jitter from qualifying as multiple events.

**Bi-Stable:** sets if the observed trigger is bi-stable or not. (If the value can be triggered in either direction.) Some profiles require this to be set a certain way. (0 = Not Bi-Stable, 1 = Bi-Stable)

**Push Profile Config 1:** determines if the values saved are sent to the sensor. In almost every case, if changes are made you will need to set this to true. The only exception would be if you are working between multiple systems such as iMonnit portal and iMonnit Enterprise. If you are moving a sensor from one platform to the other it may be useful to set these values in the new platform to match the old platform without needing them to be sent to the sensor.
**Example**

We’ll outline this general use with an example for configuring a water sensor. Other sensor profiles that are also trigger category sensors will be similar but refer to individual documentation as some values may be used differently in different profiles.

**Event Detection Type**: 2 = Change (this will cause the sensor to report aware when water is introduced, and when water is removed)

**Event Detection Count**: 30 Default filter value for water sensor

**Event Detection Period**: 50 Default filter value for water sensor

**Rearm Time**: one second once water is detected. It will wait for 1 one second before checking to see if the water is still present.

**Bi-Stable**: 1 = Bi-Stable - the sensor will monitor both the arrival and the departure of water and immediately report both states.

**Push Profile Config 1**: True will send the modified configuration to the sensor.

Assumptions for example:

SensorID = 1234

AuthToken = ABCDEFG

Full URI built to send our example:

https://www.imonnit.com/xml/SensorSetCalibration/ABCDEFG?sensorID=1234&calibration1=4294967295&calibration2=4294967295&calibration3=4294967295&calibration4=4294967295&eventDetectionType=2&eventDetectionCount=30&eventDetectionPeriod=50&rearmTime=1&biStable=1&pushProfileConfig1=true&pushProfileConfig2=false&pushAutoCalibrateCommand=false

**General Measurement Sensor Configurations**

Sensors in the Measurement category generally have more complex readings to acquire and therefore have more required values and are more specific in their configurations. Measurement sensors typically do utilize the threshold portion of the sensor configurations so the API call to configure those values will be covered in the second half of this section. In the “SensorSetCalibration” endpoint there are a number of parameters that you must send. Even though some of these values are not utilized by the Measurement category, all fields are required to be populated. Best practice is to call the “SensorGetCalibration” endpoint and start by populating all the fields with their existing value. Then modifying the fields you wish to change. In the following list of parameters the starred (*) fields are fields that are forwarded to the Measurement category of sensors.

**Sensor Set Calibration Parameter List:**
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Type</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensorID:</td>
<td>Integer</td>
<td>Unique identifier of the sensor</td>
</tr>
<tr>
<td>*calibration1:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>*calibration2:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>*calibration3:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>*calibration4:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>eventDetectionType:</td>
<td>Integer</td>
<td>Type of event detected</td>
</tr>
<tr>
<td>eventDetectionCount:</td>
<td>Integer</td>
<td>Number of events required to trigger</td>
</tr>
<tr>
<td>eventDetectionPeriod:</td>
<td>Integer</td>
<td>Time window for event count to be reached</td>
</tr>
<tr>
<td>rearmTime:</td>
<td>Integer</td>
<td>Time before event can be triggered again</td>
</tr>
<tr>
<td>biStable:</td>
<td>Integer</td>
<td>Direction of event</td>
</tr>
<tr>
<td>*pushProfileConfig1:</td>
<td>boolean</td>
<td>Set the configuration page to be pushed to the sensor</td>
</tr>
<tr>
<td>*pushProfileConfig2:</td>
<td>boolean</td>
<td>Set the configuration page to be pushed to the sensor</td>
</tr>
<tr>
<td>*pushAutoCalibrateCommand:</td>
<td>boolean</td>
<td>Set the auto calibrate command to be pushed to the sensor</td>
</tr>
</tbody>
</table>

Calibration 1: field used to hold calibration or other configurable parameters used by the sensor firmware to acquire accurate readings and or control sensor state.

Calibration 2: field used to hold calibration or other configurable parameters used by the sensor firmware to acquire accurate readings and or control sensor state.

Calibration 3: field used to hold calibration or other configurable parameters used by the sensor firmware to acquire accurate readings and or control sensor state.

Calibration 4: field used to hold calibration or other configurable parameters used by the sensor firmware to acquire accurate readings and or control sensor state.

Push Profile Config 1: determines if the values saved are sent to the sensor. In almost every case when you make changes you will need to set this to true. The only exception would be if you are working between multiple systems such as iMonnit portal and iMonnit Enterprise. If you are moving a sensor from one platform to the other it may be useful to set these values in the new platform to match the old platform without needing them to be sent to the sensor. In measurement sensors this field determines if the Threshold page (defined below) will be sent to the sensor or not.

Push Profile Config 2: determines if the values saved are sent to the sensor. In almost every case when you make changes you will need to set this to true. The only exception would be if you are working between multiple systems such as iMonnit portal and iMonnit Enterprise. If you are moving a sensor from one platform to the other it may be useful to set these values in the new platform to match the old platform without needing them to be sent to the sensor. In measurement sensors this field determines if the Calibration page (defined here) will be sent to the sensor or not.
**Push Auto Calibrate Command:** sets a specific Action Control Command to the sensor often used to calibrate the sensor. There is no general use of this field it is always profile specific and not used in all profiles.

**Example**

We’ll outline one general use with an example for configuring a temperature sensor back to default calibrations. Because this profile will modify these values internally when performing a calibration, it is not advised to manually modify these values except to set back to default values if required. Other sensor profiles that are also Measurement category sensors may be similar but refer to individual documentation as values will be used differently in different profiles.

**Calibration 1:** 0 default value for temperature sensor.

**Calibration 2:** 100000 default value for temperature sensor.

**Calibration 3:** 560000 default value for temperature sensor.

**Calibration 4:** 10000 default value for temperature sensor.

**Push Profile Config 1:** False, this will cause the thresholds stored in the database NOT to be sent to the sensor.

**Push Profile Config 2:** True, this will cause these default values to be sent to the sensor.

**Push Auto Calibrate Command:** False, action control command will not be sent to the sensor

Assumptions for example:

- SensorID = 1234
- AuthToken = ABCDEFG

Full URI built to send our example:

https://www.imonnit.com/xml/SensorSetCalibration/ABCDEFG?sensorID=1234&calibration1=0&calibration2=100000&calibration3=560000&calibration4=10000&eventDetectionType=-2147483648&eventDetectionCount=-2147483648&eventDetectionPeriod=-2147483648&rearmTime=0&biStable=-2147483648&pushProfileConfig1=true&pushProfileConfig2=false&pushAutoCalibrateCommand=false

**Sensor Set Threshold Parameter List:**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Type</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensorID:</td>
<td>Integer</td>
<td>Unique identifier of the sensor</td>
</tr>
</tbody>
</table>
measurementsPerTransmission: Integer
Number of times per heartbeat the thresholds are checked.

minimumThreshold: Integer
Minimum Threshold

maximumThreshold: Integer
Maximum Threshold

hysteresis: Integer
Hysteresis applied before entering normal operation mode

**Measurements per Transmission**: integer defining the frequency of comparing against minimum and maximum thresholds to determine if the sensor should enter or exit the aware state. Divides the heartbeat into this many measurements so actual frequency is determined by dividing the current heartbeat by this values. Minimum assessment frequency is once per heartbeat. Maximum assessment frequency is heartbeat/this value or 1 second whichever is higher. Example: 60 min heartbeat and 30 measurements per transmission will result in the sensor comparing actual reading against thresholds every 2 minutes.

**Minimum Threshold**: if measured reading is observed below this value the sensor will enter the Aware State.

**Maximum Threshold**: if measured reading is observed above this value the sensor will enter the Aware State.

**Hysteresis**: for the sensor to go from the Aware State to the Standard State it must re-enter the configured thresholds by this margin before it will return to the standard state. This prevents the sensor from oscillating between Standard and Aware States when observed readings are right at the configured threshold value.

**Example**

We’ll outline one general use with an example for configuring a temperature sensor for generic refrigeration values. The temperature sensor profile configures values and degrees Celsius *10. This allows for the configuration to reserve one decimal place yet still pass values as integers. Assuming the default heartbeat of 120 minutes, setting Measurements per Transmission to 240 will configure the sensor to check against thresholds every 30 seconds. In this example our minimum threshold is 0 deg C (freezing 32 deg F) and our maximum threshold is 4.5 deg C (40 deg F). Other sensor profiles that are also Measurement category sensors may be similar but refer to individual documentation as values will be used differently in different profiles.

**Measurements per Transmission**: 240

**Minimum Threshold**: 0

**Maximum Threshold**: 45

**Hysteresis**: 1

Assumptions for example:
SensorID = 1234
AuthToken = ABCDEFG

Full URI built to send our example:

https://www.imonnit.com/xml/SensorSetThreshold/ABCDEFG?sensorID=1234&measurementsPerTransmission=240&minimumThreshold=0&maximumThreshold=45&hysteresis=1
**Water Detection Sensor**

This document outlines the profile specific values that can be configured using the iMonnit Rest API. Each sensor profile has a unique numeric application identifier associated with it. Certain places in the API this ApplicationID is presented. For the Water Detection profile the identifier is 4. This application is a member of the Trigger Category of sensors.

**Sensor Set Calibration Endpoint**

In the “SensorSetCalibration” endpoint there are a number of parameters that you must send. Even though some of these values are not utilized by the Water Detection Profile, all fields are required to be populated. Best practice is to call the “SensorGetCalibration” endpoint and start by populating all the fields with their existing value. Then modifying the fields you wish to change. In the following list of parameters the starred (*) fields are fields that are forwarded to the Water Detection profile.

**Sensor Set Calibration Parameter List:**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Type</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensorID:</td>
<td>Integer</td>
<td>Unique identifier of the sensor</td>
</tr>
<tr>
<td>calibration1:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>calibration2:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>calibration3:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>calibration4:</td>
<td>64 bit Integer</td>
<td>Value used to store calibration values on sensors.</td>
</tr>
<tr>
<td>*eventDetectionType:</td>
<td>Integer</td>
<td>Type of event detected</td>
</tr>
<tr>
<td>*eventDetectionCount:</td>
<td>Integer</td>
<td>Number of events required to trigger</td>
</tr>
<tr>
<td>*eventDetectionPeriod:</td>
<td>Integer</td>
<td>Time window for event count to be reached</td>
</tr>
<tr>
<td>*rearmTime:</td>
<td>Integer</td>
<td>Time before event can be triggered again</td>
</tr>
<tr>
<td>*biStable:</td>
<td>Integer</td>
<td>Direction of event</td>
</tr>
<tr>
<td>*pushProfileConfig1:</td>
<td>boolean</td>
<td>Set the configuration page to be pushed to the sensor</td>
</tr>
<tr>
<td>pushProfileConfig2:</td>
<td>boolean</td>
<td>Set the configuration page to be pushed to the sensor</td>
</tr>
<tr>
<td>pushAutoCalibrateCommand:</td>
<td>boolean</td>
<td>Set the auto calibrate command to be pushed to the sensor</td>
</tr>
</tbody>
</table>

**Event Detection Type:** determines which value the sensor will report as its Aware State. Sensors prior to version 2.3.0.0 only allow two values (Water Present/ Water Not Present). Starting with version 2.3.0.0 trigger sensors now support three options (Water Present/ Water Not Present / Change). Change means that if the reading is different than the previous reading it will be marked Aware.

**Event Detection Count:** this field is utilized with the Event Detection Period parameter to create a software filter to adjust sensitivity and prevent false triggers. This is the number of readings the processor must reach within the detection period before it is qualified as an actual event. Each profile counts event differently, check profile specific documentation before modifying this value.

**Event Detection Period:** this field is utilized with the Event Detection Count parameter to create a software filter to adjust sensitivity and prevent false triggers. This is the time window the processor
uses to observe readings before it is qualified as an actual event. Each profile counts event differently, check profile specific documentation before modifying this value.

**Rearm Time:** once an event is qualified and triggered this is the amount of time in seconds the sensor is deactivated to prevent jitter from qualifying as multiple events.

**Bi-Stable:** sets if the observed trigger is bi-stable or not. (If the value can be triggered in either direction.) Some profiles require this to be set a certain way. (0 = Not Bi-Stable, 1 = Bi-Stable)

**Push Profile Config 1:** determines if the values saved are sent to the sensor. In almost every case when you make changes you will need to set this to true. The only exception would be if you are working between multiple systems such as iMonnit portal and iMonnit Enterprise. If you are moving a sensor from one platform to the other it may be useful to set these values in the new platform to match the old platform without needing them to be sent to the sensor.

**Permitted Values/Ranges**

**Event Detection Type:**
- 0 = Present (this will cause the sensor to report aware when water is detected)
- 1 = Not Present (this will cause the sensor to report aware when water is not detected)
- 2 = Change (this will cause the sensor to report aware when current reading is different than previous reading)

**Event Detection Count:** observed instances of water detected

1-65535

**Event Detection Period:** milliseconds

1-65535

**Rearm Time:** seconds

1-65535

**Bi-Stable:** detects and transmits reading for any change in reading

0 = Not Bi-Stable
1 = Bi-Stable

**Default Values**

**Event Detection Type:** 0

**Event Detection Count:** 30

**Event Detection Period:** 50
Rearm Time: 1

Bi-Stable: 1

Water Detection Example

In the following example we will set the water sensor to send Aware messages every time a change is detected in the presence of water.

Event Detection Type: 2 = change (this will cause the sensor to report aware when water is introduced, and when water is removed)

Event Detection Count: 30 default filter value for water sensor

Event Detection Period: 50 default filter value for water sensor

Rearm Time: one second once water is detected it will wait for one second before checking to see if the water is still present.

Bi-Stable: 1 = Bi-Stable the sensor will monitor both the arrival and the departure of water and immediately report both states.

Push Profile Config 1: True will send the modified configuration to the sensor.

Assumptions for example:

SensorID = 1234

AuthToken = ABCDEFG

Full URI built to send our example:

https://www.imonnit.com/xml/SensorSetCalibration/ABCDEFG?sensorID=1234&calibration1=4294967295&calibration2=4294967295&calibration3=4294967295&calibration4=4294967295&eventDetectionType=2&eventDetectionCount=30&eventDetectionPeriod=50&rearmTime=1&biStable=1&pushProfileConfig1=true&pushProfileConfig2=false&pushAutoCalibrateCommand=false