Magnetometer

W3C Working Draft, 27 November 2023

▼ More details about this document

This version:
https://www.w3.org/TR/2023/WD-magnetometer-20231127/

Latest published version:
https://www.w3.org/TR/magnetometer/

Editor's Draft:
https://w3c.github.io/magnetometer/

Previous Versions:
https://www.w3.org/TR/2023/WD-magnetometer-20231026/

History:
https://www.w3.org/standards/history/magnetometer/

Feedback:
public-device-apis@w3.org with subject line “`[magnetometer]` *… message topic …*” (archives)
Magnetometer Issues Repository

Editors:
Anssi Kostiainen (Intel Corporation)
Rijubrata Bhaumik (Intel Corporation)

Test Suite:
web-platform-tests on GitHub

Abstract

This specification defines a concrete sensor interface to measure magnetic field in the X, Y and Z axis.

Status of this document

This section describes the status of this document at the time of its publication. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

This document was published by the Devices and Sensors Working Group as a Working Draft using the Recommendation track. This document is intended to become a W3C Recommendation.

If you wish to make comments regarding this document, please send them to public-device-apis@w3.org (subscribe, archives). When sending e-mail, please put the text “magnetometer” in the subject, preferably like this: “[magnetometer] …*summary of comment*…”. All comments are welcome.

Publication as a Working Draft does not imply endorsement by W3C and its Members. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

This document is governed by the 03 November 2023 W3C Process Document.

This specification is looking for developer feedback and high value use cases. Please provide your feedback via GitHub.

This document is maintained and updated at any time. Some parts of this document are work in progress.

Table of Contents

§ 1. Introduction

Magnetometer extends the Generic Sensor API [GENERIC-SENSOR] to provide information about the magnetic field as detected by the device's primary magnetometer sensor. The magnetometer sensor measures the magnetic field for all three physical axes (x, y, z) in μT (micro Tesla).

This specification defines two new interfaces:

- `Magnetometer` that reports calibrated magnetic field values, and
- `UncalibratedMagnetometer` that reports uncalibrated magnetic field values.

The ***magnetic field*** is a field that exerts magnetic force on magnetometer sensor due to the magnetic effect generated by electric currents, magnetic materials or Earth's magnetic force that is attributed to the combined effects of the planetary rotation and the movement of molten iron in the Earth's core.

Hard iron distortion is created by objects that produce a magnetic field, such as magnetized iron.

Soft iron distortion stretches or distorts the magnetic field and is caused by metals such as nickel and iron.

The ***calibrated magnetic field*** is a magnetic field with hard iron distortion and soft iron distortion correction applied.

The ***uncalibrated magnetic field*** is the magnetic field without hard iron distortion correction and with soft iron distortion correction applied, and as such reports changes in the magnetic field caused by magnetized objects moving near the magnetometer.

§ 2. Examples

EXAMPLE 1 ¶

```
let sensor = new Magnetometer();
sensor.start();

sensor.onreading = () => {
    console.log("Magnetic field along the X-axis " + sensor.x);
    console.log("Magnetic field along the Y-axis " + sensor.y);
    console.log("Magnetic field along the Z-axis " + sensor.z);
};

sensor.onerror = event => console.log(event.error.name, event.error.m
```

§ 3. Security and Privacy Considerations

Magnetometer provides information about magnetic field, and in theory, can expose location of a user. For example, attack vector could be pre-magnetized surface in a particular location, or mapping between location and constant magnetic field disturbances caused by the building. Due to non-uniform strength of the Earth's magnetic field, another attack vector could be exposure or validation of the user's location. For example, if the end user is connected through VPN, magnetic field associated with geo IP information can be compared with magnetometer readings at real location, therefore, tell whether user is using VPN or not. Implementors should be aware of potential risk of side-channel leaks via the correlations of magnetic field strength and other aspects such as CPU execution, which under certain circumstances may potentially leak the information about used applications or websites visited in other tabs. [MAGSPY]

Uncalibrated magnetometer readings could be affected by magnetized objects nearby, such as jewelry, thereby exposing information that might be used for keystroke monitoring.

To mitigate these specific threats, user agents should use one or both of the following mitigation strategies:

- limit maximum sampling frequency
- reduce accuracy of sensor readings

These mitigation strategies complement the generic mitigations defined in the Generic Sensor API [GENERIC-SENSOR].

§ 4. Permissions Policy integration

This specification defines a policy-controlled feature identified by the string "***`magnetometer`***". Its default allowlist is `"self"`.

§ 5. Model

The ***Magnetometer*** sensor type has the following associated data:

Extension sensor interface
: `Magnetometer`

Sensor permission names
: "***`magnetometer`***"

Sensor feature names
: "magnetometer"

Permission revocation algorithm

Invoke the generic sensor permission revocation algorithm with "`magnetometer`".

Virtual sensor type

"***`magnetometer`***"

The latest reading for a `Sensor` whose sensor type is Magnetometer must include:

- Three entries whose keys are "x", "y", "z" and whose values contain magnetic field about the corresponding axes.

The ***Uncalibrated Magnetometer*** sensor type has the following associated data:

Extension sensor interface

`UncalibratedMagnetometer`

Sensor permission names

"`magnetometer`"

Sensor feature names

"magnetometer"

Permission revocation algorithm

Invoke the generic sensor permission revocation algorithm with "`magnetometer`".

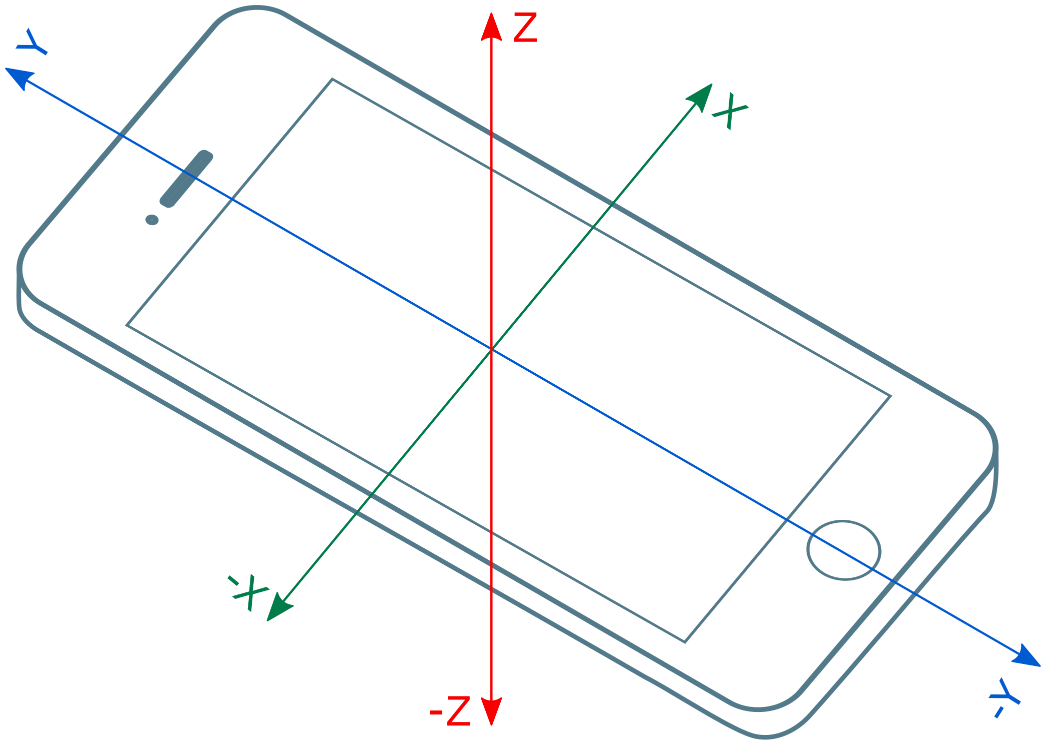
Virtual sensor type

"***`uncalibrated-magnetometer`***"

The latest reading for a `Sensor` whose sensor type is Uncalibrated Magnetometer must include:

- Three entries whose keys are "x", "y", "z" and whose values contain uncalibrated magnetic field around the 3 different axes.
- Three additional entries whose keys are "xBias", "yBias", "zBias" and whose values contain the hard iron distortion correction around the 3 different axes.

The sign of the magnetic field values must be according to the right-hand convention in a local coordinate system (see figure below).



§ 5.1. Reference Frame

The local coordinate system represents the reference frame for the `Magnetometer` and the `UncalibratedMagnetometer` readings. It can be either the device coordinate system or the screen coordinate system.

§ 6. API

§ 6.1. The Magnetometer Interface

```
[SecureContext,
  Exposed=Window]
interface Magnetometer : Sensor {
  constructor(optional MagnetometerSensorOptions sensorOptions = {});
  readonly attribute double? x;
  readonly attribute double? y;
  readonly attribute double? z;
};

enum MagnetometerLocalCoordinateSystem { "device", "screen" };

dictionary MagnetometerSensorOptions : SensorOptions {
  MagnetometerLocalCoordinateSystem referenceFrame = "device";
};
```

The ***new Magnetometer(sensorOptions)*** constructor steps are to invoke the construct a magnetometer object abstract operation with this and *sensorOptions*.

Supported sensor options for `Magnetometer` are "frequency" and "referenceFrame".

§ 6.1.1. Magnetometer.x

The `x` attribute of the `Magnetometer` interface represents the magnetic field around X-axis. In other words, this attribute returns the result of invoking get value from latest reading with `this` and "x" as arguments.

§ 6.1.2. Magnetometer.y

The `y` attribute of the `Magnetometer` interface represents the magnetic field around Y-axis. In other words, this attribute returns the result of invoking get value from latest reading with `this` and "y" as arguments.

§ 6.1.3. Magnetometer.z

The `z` attribute of the `Magnetometer` interface represents the magnetic field around Z-axis. In other words, this attribute returns the result of invoking get value from latest reading with `this` and "z" as arguments.

§ 6.2. The UncalibratedMagnetometer Interface

```
[SecureContext,
  Exposed=Window]
interface UncalibratedMagnetometer : Sensor {
  constructor(optional MagnetometerSensorOptions sensorOptions = {});
  readonly attribute double? x;
  readonly attribute double? y;
  readonly attribute double? z;
  readonly attribute double? xBias;
  readonly attribute double? yBias;
  readonly attribute double? zBias;
};
```

The ***new UncalibratedMagnetometer(sensorOptions)*** constructor steps are to invoke the construct a magnetometer object abstract operation with this and *sensorOptions*.

Supported sensor options for `UncalibratedMagnetometer` are "frequency" and "referenceFrame".

§ 6.2.1. UncalibratedMagnetometer.x

The `x` attribute of the `UncalibratedMagnetometer` interface represents the uncalibrated magnetic field around X-axis. In other words, this attribute returns the result of invoking get value from latest reading with `this` and "x" as arguments.

§ 6.2.2. UncalibratedMagnetometer.y

The `y` attribute of the `UncalibratedMagnetometer` interface represents the uncalibrated magnetic field around Y-axis. In other words, this attribute returns the result of invoking get value from latest reading with `this` and "y" as arguments.

§ 6.2.3. UncalibratedMagnetometer.z

The `z` attribute of the `UncalibratedMagnetometer` interface represents the uncalibrated magnetic field around Z-axis. In other words, this attribute returns the result of invoking get value from latest reading with `this` and "z" as arguments.

§ 6.2.4. UncalibratedMagnetometer.xBias

The `xBias` attribute of the `UncalibratedMagnetometer` interface represents the hard iron distortion correction around X-axis. In other words, this attribute returns the result of invoking get value from latest reading with `this` and "xBias" as arguments.

§ 6.2.5. UncalibratedMagnetometer.yBias

The `yBias` attribute of the `UncalibratedMagnetometer` interface represents the hard iron distortion correction around Y-axis. In other words, this attribute returns the result of invoking get value from latest reading with `this` and "yBias" as arguments.

§ 6.2.6. UncalibratedMagnetometer.zBias

The `zBias` attribute of the `UncalibratedMagnetometer` interface represents the hard iron distortion correction around Z-axis. In other words, this attribute returns the result of invoking get value from latest reading with `this` and "zBias" as arguments.

§ 7. Abstract Operations

§ 7.1. Construct a magnetometer object

input

object, a `Magnetometer` or `UncalibratedMagnetometer` object

options, a `MagnetometerSensorOptions` object.

1. Let *allowed* be the result of invoking check sensor policy-controlled features with *object*'s sensor type.
2. If *allowed* is false, then:
    1. Throw a `SecurityError` `DOMException`.
3. Invoke initialize a sensor object with *object* and *options*.
4. If *options*.`referenceFrame` is "screen", then:
    1. Set *object*'s local coordinate system to the screen coordinate system.
5. Otherwise, define *object*'s local coordinate system to the device coordinate system.

§ 8. Automation

This section extends *Generic Sensor API* § 9 Automation by providing Magnetometer-specific virtual sensor metadata.

§ 8.1. Magnetometer automation

The per-type virtual sensor metadata map must have the following entry:

key

"`magnetometer`"

value

A virtual sensor metadata whose reading parsing algorithm is parse XYZ reading.

§ 8.2. Uncalibrated Magnetometer automation

The per-type virtual sensor metadata map must have the following entry:

key

"`uncalibrated-magnetometer`"

value

A virtual sensor metadata whose reading parsing algorithm is the uncalibrated magnetometer reading parsing algorithm.

§ 8.2.1. Uncalibrated Magnetometer reading parsing algorithm

input

parameters, a JSON `Object`

output

A sensor reading or **undefined**

1. Let *reading* be the result of parse XYZ reading with *parameters*.
2. If *reading* is **undefined**.
    1. Return **undefined**.
3. Let *keys* be the list « `"xBias"`, `"yBias"`, `"zBias"` ».
4. For each *key* of *keys*
    1. Let *value* be the result of invoking parse single-value number reading with *parameters* and *key*.
        1. If *value* is **undefined**.
            1. Return **undefined**.
    2. Set *reading*[*key*] to *value*[*key*].
5. Return *reading*.

§ 9. Limitations of Magnetometer Sensors

This section is non-normative.

The direction and magnitude of the Earth's field changes with location, latitude in particular. For example, the magnitude is lowest near the equator and highest near the poles. Some hard-iron interference, meaning presence of permanent magnets (e.g. magnets in the speaker of a phone) in the vicinity of the sensor also affects the accuracy of the reading. Presence of electronic items, laptops, batteries, etc also contribute to the soft iron interference. Flight Mode option in mobile phones might help in decreasing the electro magnetic interference.

In addition to the above spatial variations of the magnetic field, time based variations, like solar winds or magnetic storms, also distort the magnetosphere or external magnetic field of the Earth.

§ 10. Use Cases and Requirements

This section is non-normative.

Magnetometers can be used for a variety of use-cases, for example:

- Sensor fusion. A common use-case for magnetometers is sensor fusion in order to generate an Absolute Orientation Sensor [MOTION-SENSORS] which is stationary to the Earth plane, or a compass, which is basically the former with corrections to the declination depending on geolocation position, such that it points to the true north. Calculating compass heading as detailed in § 11 Compass Heading Using Magnetometers.
- Virtual Reality and Augmented Reality. Magnetometer can be used to implement input using magnetic button for VR enclosures [VRBUTTON]. Head-mount tracking systems for VR and AR can use magnetometer data to help in calibration of gyroscope readings and align yaw readings with the magnetic north.
- Gesture recognition. Various interactions like writing, signing and playing an instrument can also be enabled using a magnet like a rod, pen or a ring [MAGITACT]. The user makes coarse gestures in the 3D space around the device using the magnet. Movement of the magnet affects the magnetic field sensed by the compass sensor integrated in the device. The temporal pattern of the gesture can be used as a basis for sending different interaction commands to the mobile device. Zooming, turning pages, accepting/rejecting calls, clicking items are some of the use cases.
- Indoor navigation. Navigation systems can use magnetometer data on mobile devices [MAGINDOORPOS] to detect the magnetic field inside a building. With sufficient local variability, the anomalies can be utilized in self-localization. Use cases for indoor navigation include, for example, proximity advertising, way finding in malls or airports, and geofencing.
- Metal detection. Magnetometers can be used by utility applications to detect the presence of metal nearby, e.g. finding inclusions hidden within objects.

> Requirements with respect to data coarseness and sampling frequency can vary depending on the use case at hand. For example, metal detection or input using magnetic button can likely be implemented with coarser data and using lower sampling frequency compared to gesture recognition, indoor navigation, or VR and AR use cases. In sensor fusion use cases, the sampling frequency that yeilds optimal results is dependent on e.g. sensor fusion algorithm and characteristics of other motion sensors involved.

§ 11. Compass Heading Using Magnetometers

This section is non-normative.

Compasses, instruments that align themselves with the magnetic poles of the Earth, have been used in navigation for centuries. The Earth's rotational axis defines the geographic north and south poles that we use for map references. It turns out that there is a discrepancy of around 11.5 degrees (around 1000 miles) between the geographic poles and the magnetic poles. Declination angle is applied to the magnetic direction to correct for this situation.

If the device is always level to the Earth's surface, compass heading can be determined by using just the `x` and `y` component of the Earth's magnetic field, that is, the directions planar with the Earth's surface. To determine geographic north (or true north) heading, add the appropriate declination angle.

Magnetic declination or ***declination angle*** is the angle on the horizontal plane between magnetic north and the true north and depends on the position on the Earth's surface, and changes over time. By convention, declination is positive when magnetic north is east of true north, and negative when it is to the west. You can get real time value for magnetic declination e.g. using the Magnetic declination calculator provided by the National Oceanic and Atmospheric Administration (NOAA).

The magnetic north is calculated as follows:

EXAMPLE 2 ¶

```
let sensor = new Magnetometer();
sensor.start();
let heading = Math.atan2(sensor.y, sensor.x) * (180 / Math.PI);
console.log('Heading in degrees: ' + heading);
```

The geographic north at a given latitude and longitude can be calculated as follows:

EXAMPLE 3

```
// Get the latitude and longitude, omitted for brevity here.
let latitude = 0, longitude = 0;

// Get the magnetic declination at the given latitude and longitude.
fetch('https://www.ngdc.noaa.gov/geomag-web/calculators/calculateDecl
      '?lat1=' + latitude + '&lon1=' + longitude + '&resultFormat=csv
  .then(response => response.text()).then(text => {
    let declination = parseFloat(text.replace(/^#.*$/gm, '').trim().s

    // Compensate for the magnetic declination to get the geographic
    console.log('True heading in degrees: ' + (heading + declination)
});
```

NOTE: If the device is not level to the Earth's surface, a developer needs to apply various tilt compensation techniques for which she needs a 3-axis accelerometer. Data from the orientation sensor, which is a fusion of the accelerometer and magnetometer sensors, is required to implement this particular use case.

§ 12. Acknowledgements

Tobie Langel for the work on Generic Sensor API.

§ 13. Conformance

Conformance requirements are expressed with a combination of descriptive assertions and RFC 2119 terminology. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in the normative parts of this document are to be interpreted as described in RFC 2119. However, for readability, these words do not appear in all uppercase letters in this specification.

All of the text of this specification is normative except sections explicitly marked as non-normative, examples, and notes. [RFC2119]

A ***conformant user agent*** must implement all the requirements listed in this specification that are applicable to user agents.

The IDL fragments in this specification must be interpreted as required for conforming IDL fragments, as described in the Web IDL specification. [WEBIDL]

§ Index

§ Terms defined by this specification

§ Terms defined by reference

[ACCELEROMETER] defines the following terms:

- device coordinate system
- screen coordinate system

[ECMASCRIPT] defines the following terms:

- Object

[GENERIC-SENSOR] defines the following terms:

- Sensor
- SensorOptions
- check sensor policy-controlled features
- extension sensor interface
- generic sensor permission revocation algorithm
- get value from latest reading
- initialize a sensor object
- keystroke monitoring
- latest reading
- local coordinate system
- mitigation strategies
- parse single-value number reading
- parse xyz reading
- per-type virtual sensor metadata
- reading parsing algorithm
- sensor feature names
- sensor permission name
- sensor reading
- sensor type
- supported sensor options
- virtual sensor metadata
- virtual sensor type

[INFRA] defines the following terms:

- entry
- for each
- key
- keys
- list
- map
- set
- value
- values

[MOTIONSENSORS] defines the following terms:

- absolute orientation sensor

[PERMISSIONS] defines the following terms:

- permission revocation algorithm

[PERMISSIONS-POLICY-1] defines the following terms:

- default allowlist
- policy-controlled feature

[WEBIDL] defines the following terms:

- DOMException
- Exposed
- SecureContext
- SecurityError
- double
- this
- throw

§ References

§ Normative References

[ACCELEROMETER]
Anssi Kostiainen. *Accelerometer*. 2 November 2023. CR. URL: https://www.w3.org/TR/accelerometer/

[ECMASCRIPT]
ECMAScript Language Specification. URL: https://tc39.es/ecma262/multipage/

[GENERIC-SENSOR]
Rick Waldron. *Generic Sensor API*. 22 November 2023. CR. URL: https://www.w3.org/TR/generic-sensor/

[INFRA]
Anne van Kesteren; Domenic Denicola. *Infra Standard*. Living Standard. URL: https://infra.spec.whatwg.org/

[PERMISSIONS]
Marcos Caceres; Mike Taylor. *Permissions*. 22 November 2023. WD. URL: https://www.w3.org/TR/permissions/

[PERMISSIONS-POLICY-1]
Ian Clelland. *Permissions Policy*. 17 October 2023. WD. URL: https://www.w3.org/TR/permissions-policy-1/

[RFC2119]
S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997. Best Current Practice. URL: https://datatracker.ietf.org/doc/html/rfc2119

[WEBIDL]
Edgar Chen; Timothy Gu. *Web IDL Standard*. Living Standard. URL: https://webidl.spec.whatwg.org/

§ Informative References

[MAGINDOORPOS]
Janne Haverinen, Anssi Kemppainen. *Global indoor self-localization based on the ambient magnetic field*. Informational. URL: https://doi.org/10.1016%2Fj.robot.2009.07.018

[MAGITACT]
Hamed Ketabdar, Kamer Ali Yüksel, Mehran Roshandel. *Magitact*. Informational. URL: https://dl.acm.org/doi/10.1145/1719970.1720048

[MAGSPY]
Nikolay Matyunin, Yujue Wang, Tolga Arul, Kristian Kullmann, Jakub Szefer, Stefan Katzenbeisser. *MagneticSpy: Exploiting Magnetometer in Mobile Devices for Website and*

Application Fingerprinting. Informational. URL: https://dl.acm.org/doi/abs/10.1145/3338498.3358650

[MOTION-SENSORS]
Kenneth Christiansen; Alexander Shalamov. *Motion Sensors Explainer*. 30 August 2017. NOTE. URL: https://www.w3.org/TR/motion-sensors/

[VRBUTTON]
Boris Smus. *Magnetic input for Google cardboard*. Informational. URL: https://bugs.chromium.org/p/chromium/issues/detail?id=445926

§ IDL Index

```
[SecureContext,
  Exposed=Window]
interface Magnetometer : Sensor {
  constructor(optional MagnetometerSensorOptions sensorOptions = {});
  readonly attribute double? x;
  readonly attribute double? y;
  readonly attribute double? z;
};

enum MagnetometerLocalCoordinateSystem { "device", "screen" };

dictionary MagnetometerSensorOptions : SensorOptions {
  MagnetometerLocalCoordinateSystem referenceFrame = "device";
};

[SecureContext,
  Exposed=Window]
interface UncalibratedMagnetometer : Sensor {
  constructor(optional MagnetometerSensorOptions sensorOptions = {});
  readonly attribute double? x;
  readonly attribute double? y;
  readonly attribute double? z;
  readonly attribute double? xBias;
  readonly attribute double? yBias;
  readonly attribute double? zBias;
};
```