

# C API Reference

---

**MotionNode** Version 1.2

<http://www.motionnode.com/>

Copyright © 2009 GLI Interactive LLC. All rights reserved.

The coded instructions, statements, computer programs, and/or related material (collectively the “Data”) in these files contain unpublished information proprietary to GLI Interactive LLC, which is protected by US federal copyright law and by international treaties.

The Data may not be disclosed or distributed to third parties, in whole or in part, without the prior written consent of GLI Interactive LLC.

The Data is provided “as is” without express or implied warranty, and with no claim as to its suitability for any purpose.

## Contents

<b>1 Introduction</b>	<b>2</b>
<b>2 Reference</b>	<b>2</b>
2.1 Functions	2
int mncapi_open(mncapi_stream_t type)	2
int mncapi_open_host(mncapi_stream_t type, const char * host, int port)	4
void mncapi_close(int handle)	4
int mncapi_set_blocking(int handle, int second)	4
int mncapi_sample(int handle, float * data, int data_size)	4
int mncapi_sample_int16(int handle, short * data, int data_size)	5
int mncapi_get_preview(float * data)	5
int mncapi_get_sensor(float * data)	5
int mncapi_get_raw(short * data)	5

## 1 Introduction

The MotionNode C Application Programming Interface (API) is a set of plain C functions that provides real-time access to the output of the MotionNode system. The C API is available as a standalone dynamic library suitable for use in applications such as LabVIEW and Matlab.

The C API depends on the open source MotionNode Software Development Kit (SDK) for the underlying communication and formatting systems. Refer to the **SDK Reference** document for more information.

Note that just like the SDK and the Viewer application, the C API accesses data streams from the MotionNode software service. All sensor configuration and management is handled through the User Interface or through scripting commands.

For convenience, the C API will attempt to scan for and start any available devices when a client opens a connection. To detect device arrival and removal and automatically start devices as they are plugged in use the **MotionNode Monitor** application.

## 2 Reference

The C API uses only built in types and has no external dependencies. All functions are thread safe. The library runs its own sampling loops independent of the client application. The client may choose any polling rate and even use the non-blocking read functionality to simplify integration.

For sample size, channel order, and unit information refer to the **Real-time Data Streams** section of the **SDK Reference** document.

### 2.1 Functions

**int mncapi\_open(mncapi\_stream\_t type)**

**Summary** Open a connection to a real-time data stream. Select `MNCAPI_PREVIEW`, `MNCAPI_SENSOR`, or `MNCAPI_RAW` stream type at connect time.

**Postcondition** `handle > 0`

**Parameter** type stream type identifier

**Return** positive integer valued handle for a connection resource, 0 value indicates error

```
#include <MotionNodeCAPI.h>
#include <stdio.h>

int main(int argc, char * argv[])
{
    int handle = mncapi_open(MNCAPI_SENSOR);
    if (handle > 0)
    {
        float data[9];
        if (mncapi_sample(handle, data, 9) > 0)
        {
            /* Have a sample of sensor data, print it out. */
            printf("a = [%f, %f, %f] g\n", data[0], data[1], data[2]);
            printf("m = [%f, %f, %f] uT\n", data[3], data[4], data[5]);
            printf("g = [%f, %f, %f] deg/sec\n",
                data[6], data[7], data[8]);
            printf("\n");
        }

        mncapi_close(handle);
    }

    return 0;
}
```

Example 1: API example usage. Connect, read calibrated sensor data, and then close.

```
int mncapi_open_host(mncapi_stream_t type,  
                    const char * host,  
                    int port)
```

**Summary** Same as `mncapi_open` with the addition of `host` and `port` fields.

**Postcondition** `handle > 0`

**Parameter** `type` stream type identifier  
`host` ip address string  
`port` integer port number

**Return** positive integer valued handle for a connection resource,  
0 value indicates error

```
void mncapi_close(int handle)
```

**Summary** Close an open connection to a real-time data stream.

**Parameter** `handle` integer valued handle for a connection resource,  
set to 0 to close all existing connections

```
int mncapi_set_blocking(int handle, int second)
```

**Summary** Set blocking time out or behavior for calls to `mncapi_sample` and other reading calls. The default behavior is a one second time out, `MNCAPI_DEFAULT`. To disable time out, use `MNCAPI_FOREVER`. To disable blocking and return the most recent sample immediately, use `MNCAPI_NOBLOCK`.

**Parameter** `handle` integer valued handle for a connection resource  
`second` integer valued time out value

**Return** `MNCAPI_SUCCESS` if blocking parameter set, otherwise  
`MNCAPI_FAILURE`

```
int mncapi_sample(int handle,  
                 float * data,  
                 int data_size)
```

**Summary** Read data from a real-time data stream. Requires an existing connection handle. Note that this is intended for use with the preview and sensor data streams. Use `mncapi_sample_int16` for raw data streams.

**Precondition** `data` must have at least `data_size` elements  
`data_size` must be large enough to hold at least one complete sample

**Parameter** `handle` integer valued handle for a connection resource  
`data` array to write sample data to  
`data_size` size of data array

**Return** number of devices successfully read and copied to the  
`data` buffer, otherwise `MNCAPI_FAILURE`

```
int mncapi_sample_int16(int handle,  
    short * data,  
    int data_size)
```

**Summary** Same as `mncapi_sample` except that this is intended for use with a raw data stream.

```
int mncapi_get_preview(float * data)
```

**Summary** Simplified interface using built in resource handle. Similar to `mncapi_sample(MNCAPI_PREVIEW, data, 14)`.

**Precondition** `data` must have at least 14 elements

```
int mncapi_get_sensor(float * data)
```

**Summary** Simplified interface using built in resource handle. Similar to `mncapi_sample(MNCAPI_SENSOR, data, 9)`.

**Precondition** `data` must have at least 9 elements

```
int mncapi_get_raw(short * data)
```

**Summary** Simplified interface using built in resource handle. Similar to `mncapi_sample_int16(MNCAPI_RAW, data, 9)`.

**Precondition** `data` must have at least 9 elements