BMIDI White Paper

Created
10/17/2010

Updated
10/26/2017

Authors	**Florian Bömers** **Rüdiger Pape**
Company	Bome Software
Web site	http://www.bome.com

Contents

1 About this Document....3
2 What is BMIDI?....3
3 Usage Scenarios....4
3.1 Enhance your Application....4
3.2 Add MIDI Support to your Hardware....4
3.3 Desktop MIDI Support for Mobile Apps....4
4 Windows vs. macOS....5
5 Performance....5
6 BMIDI INPUT vs. OUTPUT....5
7 The BMIDI "C" API....6
7.1 Overview....6
7.2 Code Example....6
8 The Windows Bus Driver....7
9 Windows Installer....7
10 Support....8
10.1 Code Level....8
10.2 Custom Implementations....8
10.3 Bugs/Crashes....8
10.4 Updates....8
11 Deliverables....8
12 Licensing....8
13 Branding....9
14 Licensing MT Player with BMIDI....9
15 What's New in BMIDI v2?....10
16 About Bome Software....11
16.1 About....11
16.2 Contact....11
16.3 Imprint....11

1 About this Document

This document describes the BMIDI virtual driver and the SDK, as well as licensing options.

It applies to BMIDI version 2 (“BMIDI 2”).

2 What is BMIDI?

BMIDI is a cross platform virtual MIDI driver that allows applications to expose one or more MIDI ports to other MIDI applications. Those other applications use the BMIDI ports as if they were external hardware MIDI ports.

BMIDI is not a loopback driver, because one endpoint is always “private”, i.e. in your application. That ensures best performance.

A BMIDI application uses the BMIDI “C” API to add or remove ports, to query ports, and to send/receive data to/from ports. The same API is used for Windows, macOS, and iOS. We also have a C++ MIDI device abstraction layer that is available for separate licensing, too.

Overview:

- send and receive MIDI data to/from other MIDI applications
- MIDI messages can be arbitrary length
- ports can have arbitrary names
- create unlimited number of ports
- high performance
- rock stable
- cross platform driver and simple “C” API
- 32-bit and 64-bit compatible
- in use by more than an estimated 50'000 end users
- transparent to users

Windows specific:

- silent installer on Windows (no “hardware wizard”)
- WDM kernel driver
- exposes both MME and DirectMusic ports

BMIDI ports are used, for example, in **Bome MIDI Translator Pro**, and by companies like **Native Instruments**, **TouchOSC**, **Ctrl+Console**, **Ney-Fi**, **Stanton**, and by many other OEM customers using it to relay MIDI data from their applications.

3 Usage Scenarios

3.1 Enhance your Application

If you are a software manufacturer, you may need to communicate with other software via MIDI. A clumsy way to do that is by way of loopback ports: users will need to install a separate driver, and ports are named after the loopback manufacturer, not your application.

BMIDI ports, however, allow seamless communication with other MIDI applications. The user will only need to intuitively select the MIDI port named like your application in the other application.

There is a wide variety of software applications that benefit from their own virtual ports. Examples are MIDI effects, sound engines, MIDI clock master software (to sync other music software with yours), etc.

3.2 Add MIDI Support to your Hardware

If you're a hardware manufacturer, there are numerous reasons why you would not add a MIDI port directly in your hardware driver. Examples are

- use of a proprietary protocol
- use of an OEM driver without MIDI port functionality

With BMIDI, you can add MIDI ports to your hardware, completely transparent to the user. All you need to do is create a simple user mode program which acts as a bridge from BMIDI to your hardware. To make it work like a driver, it is easy to install it as autostart in a way that the user does not see it. We have provided many such solutions to customers – please inquire if you are interested in licensing a complete BMIDI application.

3.3 Desktop MIDI Support for Mobile Apps

If you're a manufacturer of a MIDI app for mobile devices, you may want to expose a MIDI port to desktop applications. Providing a bridge with BMIDI ports fulfills this task easily, as many BMIDI customers do.

Bome Software also offers a proprietary high performance network protocol implementation (“QBMNP”) for discovery and MIDI i/o from iOS and Android apps with a desktop application on Windows, OSX and Linux. Please inquire for licensing options.

Last, but not least, Bome Software also offers the MT Player for OEM licensing, which will also provide BMIDI ports, QBMNP, a MIDI mapping runtime with all features of Bome MIDI Translator Pro, and a customizable GUI. Please inquire for more details.

4 Windows vs. macOS

Although the same source code will work on Windows and macOS, there are some architectural differences:

Windows:

- MME ports and DirectMusic ports
- WDM driver
- create and remove ports at runtime without admin privileges
- compatible with Windows XP, Vista, 7, 8, 10 (each 32-bit and 64-bit)
- signed driver, will not show driver install nag screen (except on XP).
- Uses a DLL shared with other BMIDI applications
- usually "sticky" ports, will stay even if your application is not running
- port names cannot exceed 32 characters

MacOS and iOS:

- CoreMIDI ports
- created at runtime in user space
- compatible with macOS 10.4 – 10.12 (32-bit and 64-bit)
- static library, private to your application
- ports exist only as long as your application is running
- iOS: experimental status

5 Performance

The BMIDI implementation is optimized to provide best performance, both in terms of latency and throughput. Compared to conventional loopback drivers, the architecture eliminates one path through the OS API's.

Informal benchmarks showed a round-trip delay of approx. 30 microseconds, i.e. 15 microsecs one way from a BMIDI app to a MIDI app, and vice versa.

As comparison: Class compliant USB MIDI has a typical latency of 2 milliseconds, more than 100 times the delay of BMIDI.

BMIDI's throughput is only limited by the computer's processing power. It is much faster than MIDI 1.0 speed.

We have not run any benchmarks on macOS, but the BMIDI software in production does not exhibit any performance problems.

6 BMIDI INPUT vs. OUTPUT

The "direction" of a BMIDI port is from point of view of other applications: the BMIDI INPUT ports are seen by other applications as MIDI INPUT ports. They receive the MIDI data by way of the OS API's. On the other end, the "private" end, is the BMIDI API, which is used to send MIDI data to that virtual port.

Conversely, BMIDI OUTPUT ports act the other way round: other applications open them as OUTPUT port and can send MIDI data to it. The BMIDI application receives the data by way of the BMIDI API.

Last, but not least, there is also a IN/OUT port type which has both directions.

7 The BMIDI "C" API

7.1 Overview

The API provides very simple C functions prefixed BMIDI_ . For example, there is BMIDI_GetPortCount() to get the number of installed BMIDI ports.

BMIDI_AddPort(...) and BMIDI_RemovePort(...) will create and delete BMIDI ports.

You start using a particular port with BMIDI_Connect(...). Once you're done, release it with BMIDI_Disconnect(...).

To send data to a BMIDI INPUT port, use BMIDI_Send(...) which takes a byte pointer and a length parameter. Message received on a BMIDI OUTPUT port are sent to your application by way of a callback function.

We can send you the header file (bmidilib2.h) after execution of a Non Disclosure Agreement (NDA).

7.2 Code Example

The following code example creates a virtual IN/OUT port and sends a MIDI message to it. It waits for 10 seconds during which it prints whenever a message is sent to it from another application. After that, the program disconnects from the virtual port and removes it from the system.

Note: for readability, this code does not do any error checking.

```
#include <bmidilib2.h>

// callback function: is called for every MIDI message received
// on a BMIDI_OUTPUT or BMIDI_INOUT port.
void BMIDI_CALLBACK_DECL onBMIDI_Data(BMIDI_HANDLE handle,
                                      BByte* msg, BInt32 len,
                                      void* userData)
{
   printf("BMIDI: received %d bytes.\n", len);
}

int main(int argc, char* argv[])
{
   BInt32 portIndex;
   BMIDI_HANDLE handle = NULL;

   // create a port and connect to it
   BMIDI_AddPort("TestAppID", "Test App Input", "Test App Output", &portIndex);
   BMIDI_Connect("TestAppID", portIndex, BMIDI_INOUT, onBMIDI_Data, NULL, &handle);

   // send a note for 200 milliseconds
   BByte noteOn[] = { 0x90, 0x40, 0x7F };
   BMIDI_Send(handle, noteOn, 3);
   sleepMillis(200);
   BByte noteOff[] = { 0x80, 0x40, 0x40 };
   BMIDI_Send(handle, noteOff, 3);

   // wait 5 seconds
   sleepMillis(5000);

   // disconnect and remove the port
   BMIDI_Disconnect(handle, BMIDI_INOUT);
   BMIDI_RemovePort("TestAppID", portIndex);
   return 0;
}
```

8 The Windows Bus Driver

On Windows, the BMIDI bus driver needs to be installed (with admin rights) once. This bus driver is then responsible for adding/removing ports from the BMIDI API without necessity of admin rights or UAC prompt. The bus driver is shared from all BMIDI apps.

With the BMIDI SDK, we provide a binary installer program, which installs the bus driver and is typically run from your application installer. That installer can be run in “silent” mode, so your users won't see that it's a separate installer.

On macOS, there is no need for the bus driver: you do not need to install anything.

9 Windows Installer

On Windows, you should use our provided installer program to install the bus driver.

1) Silent Install Program
You can run the separate BMIDI installer program with certain parameters so that it does not show the installer user interface (with “Next” buttons, ...).

2) Avoiding "detected new hardware" wizards
On XP, you will always get the wizard at the time of port installation. On Vista and later, our installer applies a trick to not show such hardware messages for the Bome ports (during installation, there is one UAC prompt, though, if enabled).

10 Support

10.1 Code Level

As part of licensing BMIDI, you get support for using the API in your particular implementation, and for trouble shooting (on code level without actually writing code). We do not provide end-user support.

10.2 Custom Implementations

We can do additional programming tasks for you, where we bill our hourly rate. Please ask for details.

10.3 Bugs/Crashes

We are a registered driver developer with Microsoft, so if your customers experience a crash, we get notified (user needs to click on “Send Report”) and we will get debugging information for fixing the issue.

Note that the BMIDI driver is bug free since 2007 (there had been installer problems, which are fixed in v2).

10.4 Updates

The BMIDI license covers free updates for this driver architecture and for the current supported OS versions each. We cannot guarantee compatibility with future OS versions, though we will most likely provide updates.

11 Deliverables

- bmidilib2.h header file
- dynamic library (.dll), static library (.lib / .a)
- Windows: driver installer
- test project
- bmidi_string: UTF-8 conversion package (in form of source code)

12 Licensing

Licensing BMIDI will give you full redistribution rights of the compiled binary BMIDI code in your products. So you are not only allowed to use the driver and build your software with it, but you can also redistribute the licensed parts to your customers. The redistribution rights are non-revocable (that's for your protection).

One license covers Windows, macOS, and iOS.

For licensing, there is a one-time license fee. We're also open to handling part of the license fee by way of royalties, please ask us.

13 Branding

The ports can be freely named, which covers most branding needs.

However, on Windows, the driver will indicate “Bome Software” in the version info and in the digital certificate. The bus driver is shared with other BMIDI installations.

If you want the driver to be completely independent from the Bome driver (i.e. it's possible to install Bome's bus driver and your bus driver side by side), and have your name in version info, a higher license fee applies (because it requires custom coding and testing). Additionally, you will need to register your own Verisign code certificate ($200-$400 per year). Other code certificates will not work.

14 Licensing MT Player with BMIDI

For many applications, you may not need to reinvent the wheel: we can license a custom version of our MT Player application to you. It provides freely named BMIDI ports, a powerful MIDI mapping engine (from Bome MIDI Translator Pro) and network MIDI support (“QBMNP”). It runs on Windows, macOS, Linux, iOS, and Android, and has many customization options regarding GUI and installation. For example, it can be configured to start automatically with the system and be invisible (like a driver), or just display a task bar icon. Please ask us for more details.

15 What's New in BMIDI v2?

Compared to BMIDI 1:

General:

- 32-bit and 64-bit library
- increased performance
- simplified API *)
- product name change: “Bome Virtual MIDI”
- full backwards compatibility with BMIDI 1

Windows:

- no more port name prefixing in MME applications (“2-” prefix)
- Device Manager shows the name of each port
- improved Windows installer:
  - fixed error 14 during installation
  - better compatibility with Windows 8 and later
  - consistent install location
  - detect running applications and close them prior to installation
- updating/reinstalling the driver will not remove any existing BMIDI ports

iOS:

- iOS compatibility (experimental)

***) Simplified API:**

- your ports are now logically separated from other apps' ports
- no need to manually create unique PortID's – ports are identified by name
- no need to create a thread for receiving data, the library sends the MIDI data to a callback in your application
- simplified port discovery and API usage
- with BMIDI 2, our own application layer needs 40% less code

16 About Bome Software

16.1 About

Bome™ Software is a small team of professional software developers creating powerful applications since 1996. The success of the works of Florian Bömers lead to the formation of Bome Software and the ongoing development of innovative music software and development tools.

Bome Software has developed and released MIDI and audio software for Windows, macOS, iOS, and Linux. 2015 marks a new milestone with the introduction of the first hardware product, the BomeBox™: a versatile USB, MIDI, and Ethernet connector and processor device.

As a small and flexible team, we can offer individual services for your specific needs. We are dedicated to continuous development and provide qualified support beyond anonymous telephone hotlines. Thousands of satisfied customers worldwide — ranging from individuals, music studios, theaters, governments, to Fortune 500 companies and Oscar winners — already benefit from Bome's products and services. Please see some customer testimonials.

Bome Software is an active member of the MIDI Manufacturers Association, participating in the standardization process of the next generation of MIDI.

16.2 Contact

Bome Software GmbH & Co. KG
Florian Bömers
Petra-Kelly-Str. 15
80797 München, Germany
Tel: +49 (0)89 45219247
Fax: +49 (0)89 45219248
Email: www.bome.com/contact

16.3 Imprint

Bome Software GmbH & Co KG
Petra-Kelly-Str. 15
80797 München, Germany
Registration Court/Amtsgericht: München HRA95502
Steuernummer: 144/236/91043 Finanzamt München Abt. I
VAT ID / USt.-Id-Nr.: DE271182542

Liability held by/Gesellschafterin:
Bome Komplementär GmbH
CEO/Geschäftsführung: Florian Bömers
Registration Court/Amtsgericht: München HRB185574