# *Software Synthesizer MIDI Player /Driver Library Specification*

***Version 3.0***

**CONFIDENTIAL**

***bismark***

## History:

| Date | Version | Description |
|---|---|---|
| 2013/10/17 | 2.8 | |
| 2014/08/05 | 3.0 | Fixed data type for BSMP_CTRL_SET_SAMPLE_RATE / BSMD_CTRL_SET_SAMPLE_RATE |

目次

## 1. About This Document

This document defines the specification of the Software Synthesizer MIDI Player / MIDI Driver Library.

## 2. Abstract

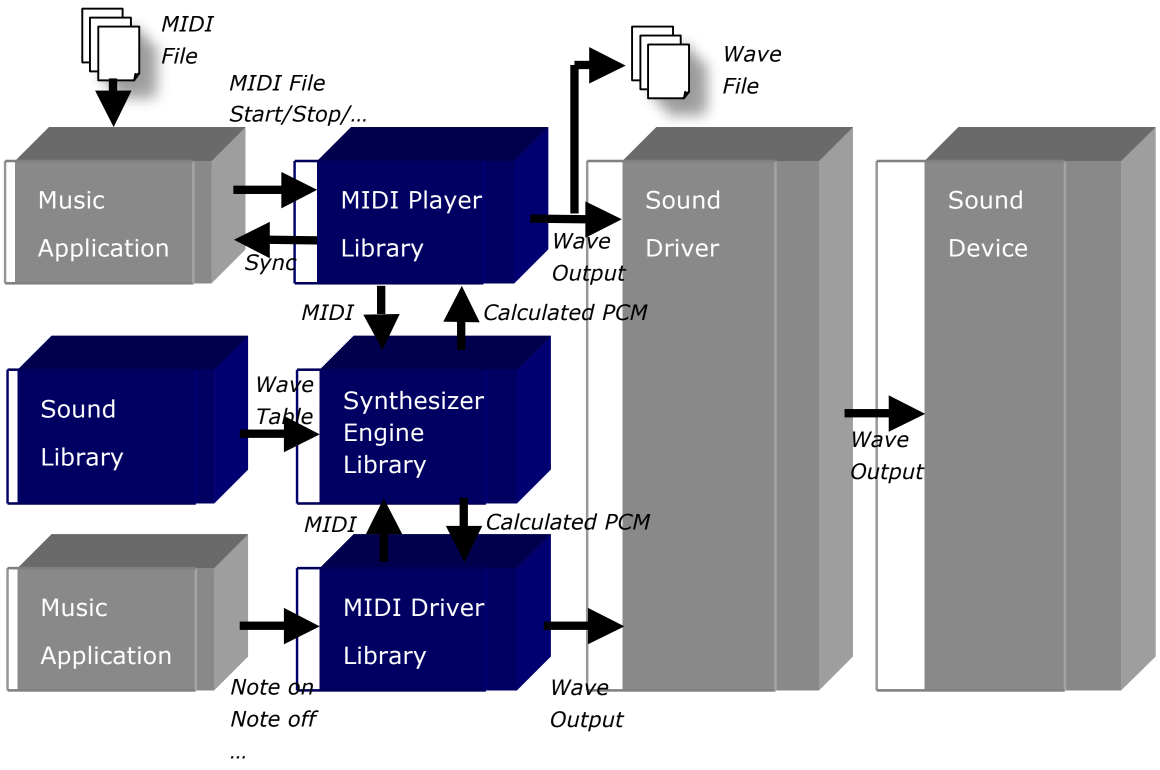
This library include Synthesizer Engine Library (bsse: bismark Synthesizer Engine), and Sound Library, also offers application interfaces for MIDI Player (bsmp: described later), and MIDI Driver (bsmd: described later).

bsmp (bismark MIDI Player) library is an additional library for Synthesizer Engine Library. It provides functions to construct MIDI file players, Karaoke players, MIDI to Wave converts easily.
The main basic functions of bsmp library are follows;

- Import MIDI files
  - Supporting SMF (Standard MIDI File)
  - Also can be added the user specified file formats as customization
- MIDI to Wave conversion using Synthesizer Engine Library
  - Including wave output device and thread schedule control for various OS
  - Export to wave files
- Application support
  - API for playback start, stop
  - Callback functions for sending synchronizing information to the application

bsmd (bismark MIDI Driver) library is an another additional library for Synthesizer Engine Library. It enables the substitution of hardware MIDI modules, and provides Real-time MIDI function and simple MIDI file player for virtual musical instrument applications.
The main basic functions of bsmd library are follows;。

- Real-time MIDI
  - Including wave output device and thread schedule control for various OS
- Simple MIDI file player
  - Supporting SMF (Standard MIDI File)



bsmp and bsmd library can not be used at the same time.

## 2.1. Supported OS

- Microsoft Windows
  - MBCS build
  - UNICODE build
- Linux/BSD
- Mac OS X
- iOS (iOS 7 SDK, armv7/armv7s)
- Android (NDK r7b)

## 2.2. Inputs

### 2.2.1. MIDI Files

- SMF (Standard MIDI File)
  - Format: 0 or 1
  - Number of track: Up to 64
  - Division / TPQN: No limitation
  - File extension: *.mid

### 2.2.2. Sound Library Files

- SoundFont
  - Version 2
  - File extension: *.sf2
- DLS (Downloadable Sounds)[1]
  - Level1, Level2, Mobile DLS
  - File extension: *.dls

## 2.3. Outputs

### 2.3.1. Wave Output Devices

- Win:
  - MME drivers
  - Steinberg ASIO 2.1 drivers (Only bsmd driver, 44100Hz sample rate)

[1] There are some limitations for supporting DLS specification. Please refer to 5.1 About DLS File Format

- Linux:
  - OSS
  - ALSA
- Mac OS X / iOS:
  - AudioQueue
  - AudioUnit (Only bsmd driver)
- Android
  - OpenSL ES

- Playback sample rate: Depends on each wave output drivers

### 2.3.2. Wave Files

bsmp library only.

- Microsoft RIFF Wave
- Apple AIFF
  - Playback sample rate: No limitation
  - Output bit depth: 16[bit]
  - Number of output channels: 2 (Interleaved)

## 2.4. File Lists

- Common
  - bsmd.h : bsmd (MIDI Driver Library) header file
  - bsmp.h : bsmp (MIDI Player Library) header file
- Win (DLL / Shared library)
  - bsmpd*.dll : Shared library
  - bsmpd*.lib : Library module
- Linux / Mac OS X / iOS / Android (Static library)
  - libbsmpd*.a (MIDI Player / MIDI Driver Library)
  - libbsmp*.a (MIDI Player Library)
  - libbsmd*.a (MIDI Driver Library)

## 2.5. Related Libraries

- Synthesizer Engine Library
  - Win
    - Included
  - Linux / Mac OS X / iOS / Android
    - libbsse*.a: Static library

# 3. MIDI Player Library Specification

## 3.1. Constants

### 3.1.1. BSMP_ERR

typedef enum for result code.

| code | description | |
|---|---|---|
| BSMP_OK | Success | |
| BSMP_ERR_PROTECTION | Protection error | |
| BSMP_ERR_INVALID_HANDLE | Invalid handle error | |
| BSMP_ERR_FILE | File error | |
| BSMP_ERR_MEMORY | Memory error | |
| BSMP_ERR_RESOURCE | Resource error | |
| BSMP_ERR_PARAM | Parameter error | |
| BSMP_ERR_AUDIO_DRIVER | Wave output error | |
| BSMP_ERR_DATA | Data error | |
| BSMP_ERR_MODULE | External module error | |
| BSMP_ERR_NOT_SUPPORTED | Unsupported error | |
| BSMP_ERR_UNDEFINED | Undefined | |

### 3.1.2. BSMP_CTRL

typedef enum for control API. Please refer to section 3.4.21 ctrl.

### 3.1.3. BSMP_CALLBACK_TYPE

typedef enum for callback types. Please refer to section 3.5 Callback (BSMP_CALLBACK).

### 3.1.4. BSMP_WAVE_FILE

typedef enum for bounced wave file formats.

| code | description | |
|---|---|---|
| BSMP_WAVE_FILE_RIFF | Microsoft RIFF Wave | |
| BSMP_WAVE_FILE_AIFF | Apple AIFF | |

### 3.1.5. BSMP_SOUND_LIBRARY_SEL_MODE

typedef enum for selection modes of sound library files.

| code | description | |
|---|---|---|
| BSMP_SOUND_LIBRARY_SEL_MODE_NORMAL | Default mode | |

## 3.2. Typedefs

### 3.2.1. BSMP_HANDLE

Handle for controlling this library.

### 3.2.2. BSMP_CALLBACK

Callback function type for sending information from this library to the user application. Please refer to section 3.5 Callback (BSMP_CALLBACK).

| *callback ()* | | |
|---|---|---|
| *Input:* | *BSMP_HANDLE handle* | *Effective handle of the library* |
| | *BSMP_CALLBACK_TYPE type* | *Callback type* |
| | *void *data* | *Pointer of the data* |
| | *void *user* | *Pointer of the specified user area* |
| *Output:* | *void* | |

### 3.2.3. BSMP_CALLBACK_BOUNCE

Callback function type for displaying progress on exporting wave files. This callback will be used on calling the API "bounce" described on section 3.4.20.

| *BSMP_CALLBACK_BOUNCE ()* | | |
|---|---|---|
| *Input:* | *int percent* | *Progress value (%)* |
| | *void *user* | *Pointer to the specified user area* |
| *Output:* | *int* | *0: Continue*<br>*1: Cancel exporting* |

### 3.2.4. BSMP_LOAD

Function type for Geting the API table（BSMP_FUNC).

## 3.3. Structures

### 3.3.1. BSMP_FUNC

Structure for API table. Please refer to section 3.4 API.

### 3.3.2. BSMP_SOUND_LIBRARY

Structure for specifying the sound library file.

```
typedef struct {
        int index; /* Index for the sound library file */
        LPCTSTR path; /* Full path of the sound library file */
} BSMP_SOUND_LIBRARY;
```

### 3.3.3. BSMP_SOUND_LIBRARY_MEMORY

Structure for specifying the sound library file mapped on the memory.

```
typedef struct {
        int index; /* Index for the sound library file */
        char *address; /* Memory address for the mapped sound library file */
        unsigned long *size; /* Size of the sound library file [Byte] */
} BSMP_SOUND_LIBRARY_MEMORY;
```

### 3.3.4. BSMP_SOUND_LIBRARY_SEL

Structure for specifying details of referring the sound library files.

```
typedef struct {
        int module; /* Module index (0, 1, ...) */
        int part; /* Part index (0, 1, ..., 15) */
        int index; /* Index of the sound library file */
        BSMP_SOUND_LIBRARY_SEL_MODE mode; /*   selection modes (section 3.1.5) */
} BSMP_SOUND_LIBRARY_SEL;
```

## 3.4. API

### 3.4.1. initialize

*BSMP_ERR initialize ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE *handle* | *Pointer of the handle (!= NULL)* |
| *BSMP_CALLBACK callback* | *Pointer of the callback function* |
| *void *user* | *Pointer of the user area for callback* |
| *void *target* | *Target independent data* |
| *const unsigned char *key* | *Key code* |

*Output:*

*Error code*

Initialize the library and Synthesizer Engine Library.

Synthesizer Engine Library loads the default sound library (from own resource, or from the defined path) to index #0.

Before using the library, the application have to call the on of initialize* () functions.

The application have to set 64 byte key code to the argument “key”.

This functions requires the fixed processing time because of loading the sound library.

The application have to set the following values to argument “target”

- Win: The handle of the parent window (HWND)
- Android: This library receives pointer of the following sturucture, and calls the Activity class method of your application using information this information.

```
typedef struct {
    JNIEnv *env;
    jobject thiz;
}
```

- Other OS: NULL

### 3.4.2. initializeWithSoundLib

*BSMP_ERR initializeWithSoundLib ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE *handle* | *Pointer of the handle (!= NULL)* |
| *BSMP_CALLBACK callback* | *Pointer of the callback function* |
| *void *user* | *Pointer of the user area for callback* |
| *LPCTSTR libraryPath* | *Full path of the sound library file* |
| *void *target* | *Target independent data* |
| *const unsigned char *key* | *Key code* |

*Output:*

*Error code*

Initialize the library and Synthesizer Engine Library.

Synthesizer Engine Library loads the sound library file on the specified path to index #0.

### 3.4.3. initializeWithSoundLibMemory

*BSMP_ERR initializeWithSoundLibMemory ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE *handle* | *Pointer of the handle (!= NULL)* |
| *BSMP_CALLBACK callback* | *Pointer of the callback function* |
| *void *user* | *Pointer of the user area for callback* |
| *char *libraryAddress* | *Address of the mapped sound library* |
| *unsigned long librarySize* | *Size of the sound library file [Byte]* |
| *void *target* | *Target independent data* |
| *const unsigned char *key* | *Key code* |

*Output:*

*Error code*

Initialize the library and Synthesizer Engine Library.

Synthesizer Engine Library loads the sound library file on the specified memory to index #0.

### 3.4.4. exit

```
BSMP_ERR exit ()
        Input:
                BSMP_HANDLE handle        Effective handle of the library
        Output:
                Error code
```

Finalize the library.

The application have to call this function before termination. If the library is playing, the application have to stop playback before calling this function.

### 3.4.5. getNumDrivers

```
int getNumDrivers ()
        Input:
                BSMP_HANDLE handle        Effective handle of the library
        Output:
                The number of supported drivers.
```

Get the number of wave output drivers supported by the library.

### 3.4.6. getNumDevices

```
int getNumDevices ()
        Input:
                BSMP_HANDLE handle        Effective handle of the library
                LPCTSTR driver            Name of wave output driver
        Output:
                The number of available wave output devices
```

Get the number of available wave output devices in the specified wave output driver.

### 3.4.7. getDriverName

*LPCTSTR getDriverName ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE handle* | *Effective handle of the library* |
| *int index* | *Index for the wave output driver* |

*Output:*

*Name of the specified wave output driver*

Get the name of the specified wave output driver.

### 3.4.8. getDeviceName

*LPCTSTR getDeviceName ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE handle* | *Effective handle of the library* |
| *LPCTSTR driver* | *Name of the wave output driver* |
| *int index* | *Index for the wave output device* |

*Output:*

*Name of the specified wave output device*

Get the name of the specified wave output device.

### 3.4.9. showDeviceControlPanel

*void showDeviceControlPanel ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE handle* | *Effective handle of the library* |
| *LPCTSTR driver* | *Name of the wave output driver* |
| *LPCTSTR device* | *Name of the wave output device* |

Display the control panes of the specified wave output device

### 3.4.10. open

*BSMP_ERR open ()*

*Input:*

| | | |
|---|---|---|
| | *BSMP_HANDLE handle* | *Effective handle of the library* |
| | *LPCTSTR driver* | *Name of the wave output driver* |
| | *LPCTSTR device* | *Name of the wave output device* |
| *Output:* | | |
| | *Error code* | |

Open the specified wave output device. If the argument "driver" and "device" is NULL, default wave output driver and device will be selected automatically.

### 3.4.11. close

*BSMP_ERR close ()*

| | | |
|---|---|---|
| *Input:* | | |
| | *BSMP_HANDLE handle* | *Effective handle of the library* |
| *Output:* | | |
| | *Error code* | |

Close the wave output device.

### 3.4.12. setFile

*BSMP_ERR setFile ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE handle* | *Effective handle of the library* |
| *LPCTSTR path* | *Full path of the MIDI file* |

*Output:*

*Error code*

Specify the MIDI sequence file with file path. See **2.2 Inputs** for available file formats.

### 3.4.13. setFileMemory

*BSMP_ERR setFileMemory ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE handle* | *Effective handle of the library* |
| *char *address* | *Memory address for the mapped MIDI file* |
| *long size* | *Size of the MIDI file [byte]* |

*Output:*

*Error code*

Specify the MIDI sequence file mapped on the memory controlled by the application.

### 3.4.14. getFileMemory

*BSMP_ERR getFileMemory ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE handle* | *Effective handle of the library* |
| *char **address* | *Pointer of the memory address* |
| *long *size* | *Pointer of the file size [byte]* |

*Output:*

*Error code*

Get the memory address and size used for loading MIDI file. This memory is controlled by the library.

### 3.4.15. getFileInfo

*BSMP_ERR getFileInfo ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE handle* | *Effective handle of the library* |
| *int *format* | *Pointer of the MIDI file format* |
| *unsigned short *division* | *Pointer of the MIDI file division [TPQN]* |
| *unsigned long *totaltick* | *Pointer of the number of tick* |
| *unsigned long *totaltime* | *Pointer of the length [s]* |

*Output:*

*Error code*

Get information of the specified MIDI sequence file.

### 3.4.16. start

| | | |
|---|---|---|
| *BSMP_ERR start ()* | | |
| *Input:* | | |
| | *BSMP_HANDLE handle* | *Effective handle of the library* |
| *Output:* | | |
| | *Error code* | |

Start playback of the specified MIDI file from current song position.

### 3.4.17. stop

| | | |
|---|---|---|
| *BSMP_ERR stop ()* | | |
| *Input:* | | |
| | *BSMP_HANDLE handle* | *Effective handle of the library* |
| *Output:* | | |
| | *Error code* | |

Stop playback of the specified MIDI file.

Calling this function means the application instructs the start of fade out process, and the playback still alive. The application has to detect the completion of the playback by the callback function described later.

Current song position will be saved after calling this function.

### 3.4.18. seek

| | | |
|---|---|---|
| *BSMP_ERR seek ()* | | |
| *Input:* | | |
| | *BSMP_HANDLE handle* | *Effective handle of the library* |
| | *unsigned long tick* | *Song Position [MIDI tick]* |
| *Output:* | | |
| | *Error code* | |

Specify song position.

### 3.4.19. isPlaying

```
int isPlaying ()
        Input:
                BSMP_HANDLE handle        Effective handle of the library
        Output:
                1: playing
                0: not playing
```

Get the flag for the library is playing the MIDI file, or not.

### 3.4.20. bounce

*BSMP_ERR bounce ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE handle* | *Effective handle of the library* |
| *LPCTSTR path* | *Full path of the output file* |
| *BSMP_WAVE_FILE type* | *Output file type* |
| *BSMP_CALLBACK_EXPORT callback* | *Callback function* |
| *void *user* | *User parameter for the callback* |

*Outpu:*

*Error code*

Outputs the result of the specified MIDI file to the wave file. This function can not be used when normal playback process is effective. (Started with 3.4.16 start)

### 3.4.21. ctrl

*BSMP_ERR ctrl ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE handle* | *Effective handle of the library* |
| *BSMP_CTRL ctrl* | *Control target* |
| *void *data* | *Address of data* |
| *int size* | *Size of data [byte]* |

*Output:*

*Error code*

Do various operations.

| ctrl | data | | description |
|---|---|---|---|
| | type | I/O | |
| BSMP_CTRL_SET_MASTER_VOLUME | int | I | Set playback volume (BSMP_VOLUME_MIN ~ BSMP_VOLUME_MAX). The default value is BSMP_VOLUME_DEF. |
| BSMP_CTRL_GET_MASTER_VOLUME | int | O | Get playback volume |
| BSMP_CTRL_SET_MASTER_KEY | int | I | Set playback key (BSMP_KEY_MIN ~ BSMP_KEY_MAX). The unit of the values is 100[cent], and the default value is BSMP_KEY_DEF. This value is not cleared on the end of the playback. |
| BSMP_CTRL_GET_MASTER_KEY | int | O | Get playback key. |
| BSMP_CTRL_SET_MASTER_TUNE | int | I | Set fine tuning (BSMP_TUNE_MIN ~ BSMP_TUNE_MAX). The unit of the values is 1[cent], and the default value is BSMP_TUNE_DEF. This value is not cleared on the end of the playback. |
| BSMP_CTRL_GET_MASTER_TUNE | int | O | Get fint tuning. |
| BSMP_CTRL_SET_SPEED | int | I | Set playback speed. (BSMP_SPEED_MIN ~ BSMP_SPEED_MAX). The unit of the value is 1[%], and the default value is BSMP_SPEED_DEF. This value is not cleared on the end of the playback. |
| BSMP_CTRL_GET_SPEED | int | O | Get playback speed. |

| ctrl | data | | description |
|---|---|---|---|
| | type | I/O | |
| BSMP_CTRL_SET_GUIDE | int | I | Set guide melody playback volume (BSMP_GUIDE_MIN ~ BSMP_GUIDE_MAX). The default value is BSMP_GUIDE_DEF. This value is not cleared on the end of the playback. |
| BSMP_CTRL_GET_GUIDE | int | O | Get guide melody playback volume. |
| BSMP_CTRL_SET_GUIDE_MAIN_CH | int | I | Set target of guide melody control.<br>-1: off<br>0: MIDI port A, MIDI channel 1<br>1: MIDI port A, MIDI channel 2<br>…<br>15: MIDI port A, MIDI channel 16<br>16: MIDI port B, MIDI channel 1<br>… |
| BSMP_CTRL_GET_GUIDE_MAIN_CH | int | O | Get target of guide melody control |
| BSMP_CTRL_SET_GUIDE_SUB _CH | int | I | Same as BSMP_CTRL_SET_GUIDE_MAIN_CH |
| BSMP_CTRL_GET_GUIDE_SUB _CH | int | O | Same as BSMP_CTRL_SET_GUIDE_MAIN_CH |

| ctrl | data | | description |
|---|---|---|---|
| | type | I/O | |
| BSMP_CTRL_SET_REVERB | *int* | I | Set effectiveness of reverb. This value is not cleared on the end of the playback. |
| BSMP_CTRL_GET_REVERB | *int* | O | Get effectiveness of reverb |
| BSMP_CTRL_GET_REVERB _AVAILABLE | *int* | O | Get availability of reverb |
| BSMP_CTRL_SET_CHORUS | *int* | I | Set effectiveness of chorus. This value is not cleared on the end of the playback. |
| BSMP_CTRL_GET_CHORUS | *int* | O | Get effectiveness of chorus |
| BSMP_CTRL_GET_CHORUS _AVAILABLE | *int* | O | Get availability of chorus |
| BSMP_CTRL_SET_DELAY | *int* | I | Set effectiveness of delay. This value is not cleared on the end of the playback. |
| BSMP_CTRL_GET_DELAY | *int* | O | Get effectiveness of delay |
| BSMP_CTRL_GET_DELAY _AVAILABLE | *int* | O | Get availability of delay |
| BSMP_CTRL_SET_REVERB_HQ | *int* | I | Set HQ Reverb (1: On, 0: Off, Customized version only) |

| ctrl | data | | description |
|---|---|---|---|
| | type | I/O | |
| BSMP_CTRL_SET_SAMPLE_RATE | int | I | Set playback sample rate [Hz] |
| BSMP_CTRL_GET_SAMPLE_RATE | int | O | Get playback sample rate [Hz] |
| BSMP_CTRL_SET_BLOCK_SIZE | long | I | Set frame size [sample] of wave output. |
| BSMP_CTRL_GET_BLOCK_SIZE | long | O | Get frame size [sample] of wave output. |
| BSMP_CTRL_SET_CHANNELS | int | I | Not supported |
| BSMP_CTRL_GET_CHANNELS | int | O | Get number of output channels |
| BSMP_CTRL_SET_POLY | int | I | Set polyphonic number of synthesizer |
| BSMP_CTRL_GET_POLY | int | O | Get polyphonic number of synthesizer |

| ctrl | data | | description |
|---|---|---|---|
| | type | I/O | |
| BSMP_CTRL_GET_SOUND_LIBRARY_NUM | int | O | Get number of the slots for sound libraries |
| BSMP_CTRL_SET_SOUND_LIBRARY | BSMP_SOUND_LIBRARY | I | Set sound library with file path |
| BSMP_CTRL_SET_SOUND_LIBRARY_MEMORY | BSMP_SOUND_LIBRARY_MEMORY | I | Set sound library with memory |
| BSMP_CTRL_SET_SOUND_LIBRARY_SEL | BSMP_SOUND_LIBRARY_SEL | I | Set selection mode for the loaded sound library |
| BSMP_CTRL_GET_SOUND_LIBRARY_SEL | BSMP_SOUND_LIBRARY_SEL | I/O | Get selection mode for the loaded sound library |
| BSMP_CTRL_SET_NO_INSTRUMENT_FIX | int | I | Set function for substituting instrument. (1: On, 0: Off) |
| BSMP_CTRL_GET_NO_INSTRUMENT_FIX | int | O | Get value for the substituting instrument. |
| BSMP_CTRL_SET_NUMBER_OF_REGIONS | int | I | Set maximum number of region in each instrument |

| ctrl | data | | description |
|---|---|---|---|
| | type | I/O | |
| BSMP_CTRL_GET_INSTRUMENT_NAME ~ BSMP_CTRL_GET_INSTRUMENT_NAME + 15 | char (TCHAR) | O | Get instrument name of the specified part (Ch1~16) |
| BSMP_CTRL_SET_MUTE ~ BSMP_CTRL_SET_MUTE + 15 | int | I | Set mute (0: Off, 1: On) to the specified part (Ch1~16) |
| BSMP_CTRL_GET_MUTE ~ BSMP_CTRL_GET_MUTE + 15 | int | O | Get mute (0: Off, 1: On) of the specified part (Ch1~16) |
| BSMP_CTRL_SET_SOLO ~ BSMP_CTRL_SET_SOLO + 15 | int | I | Set solo (0: Off, 1: On) to the specified part (Ch1~16) |
| BSMP_CTRL_GET_SOLO ~ BSMP_CTRL_GET_SOLO + 15 | int | O | Get solo (0: Off, 1: On) of the specified part (Ch1~16) |

| *ctrl* | data | | Description |
|---|---|---|---|
| | type | I/O | |
| *BSMP_CTRL_SET_CALLBACK_DELAY* | int | I | Set callback sync offset |
| *BSMP_CTRL_GET_CALLBACK_DELAY* | int | O | Get callback sync offset |
| *BSMP_CTRL_SET_PORT_SELECTION_METHOD* | int | I | Set port selection method (Customized version only) |
| *BSMP_CTRL_GET_PORT_SELECTION_METHOD* | int | O | Get port selection method (Customized version only) |
| *BSMP_CTRL_SET_WAVE* | BSMP_WAVE | I | Add wave file (customized version only) |
| *BSMP_CTRL_GET_OPEN_SL_ENGINE* | | O | Get OpenSL Engine (Android only) |
| *CTMP_CTRL_GET_OPEN_SL_ENGINE_INTERFACE* | | O | Get OpenSL Engine Interface (Android only) |
| | | | |

### 3.4.22. version

*void version ()*

*Input:*

| | |
|---|---|
| *BSMP_HANDLE handle* | *Effective handle of the library* |
| *LPTSTR engine* | *Version of Synthesizer Engine Library* |
| *int engineSize* | *Length of engine* |
| *LPTSTR player* | *Version of MIDI Player Library* |
| *int playerSize* | *Length of player* |

Get the name of MIDI Player Library and Synthesizer Engine Library.

## 3.5. Callback (BSMP_CALLBACK)

Callback function provides various information to the application. It is specified on 3.4.1 initialize, with function type defined in section 3.2.2 BSMP_CALLBACK.
This callback is not called on processing the function 3.4.20 bounce.
Each callback is called from calculation thread of synthesizer. So the application can not spend long duration on receiving them.

### 3.5.1. Open

*type = BSMP_CALLBACK_TYPE_OPEN, data = Not used*
Wave output driver has been opened

### 3.5.2. Close

*type = BSMP_CALLBACK_TYPE_CLOSE, data = Not used*
Wave output driver has been closed

### 3.5.3. Start

*type = BSMP_CALLBACK_TYPE_START, data = Not used*
Playback has been started

### 3.5.4. Stop

*type = BSMP_CALLBACK_TYPE_STOP, data = (unsigned long *) errorcode*
Playback has beed stopeed.
errorcode:

*0：Normal*

*BSMP_ERR_AUDIO_DRIVER：Error stop by wave output driver*

*BSMP_ERR_DATA：Error stop by data*

### 3.5.5. Seek

*type = BSMP_CALLBACK_TYPE_SEEK, data = Not used*

Playback song position has been changed

If your application calculates song position using 3.5.6 MIDI Clock callback, please reset song position to start, tempo to 120[BPM], on receiving this callback.

### 3.5.6. MIDI Clock

*type = BSMP_CALLBACK_TYPE_CLOCK, data = Not used*

Standard MIDI clock (24[TPQN])

### 3.5.7. Tempo

*type = BSMP_CALLBACK_TYPE_TEMPO, data = (unsigned long *) tempo*

Playback tempo has been changed ([usec/beat])

### 3.5.8. Time Signature

*type = BSMP_CALLBACK_TYPE_TIME_SIGNATURE, data = (unsigned long *) timeSignature*

Playback time signature（nn/dd/cc/bb）has been changed.

### 3.5.9. Channel Message

*type = BSMP_CALLBACK_TYPE_CHANNEL_MESSAGE, data = (unsigned long *) data*

Channel message has been sent by player

bit 31-24: MIDI Port (0x00 ~ )

bit 23 - 16: Status Byte (0x90 ~ 0xEF)

bit 15 – 8 : First Data (0x00 ~ 0x7F)

bit 7 – 0 : Second Data (0x00 ~ 0x7F)

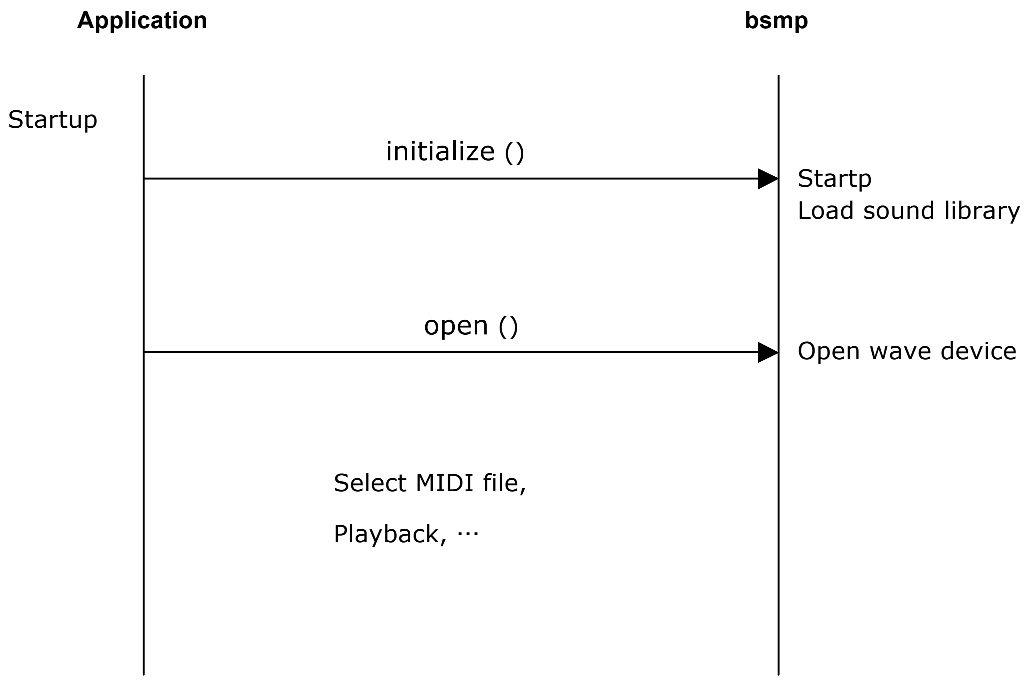
### 3.5.10. System Exclusive Message

*type = BSMP_CALLBACK_TYPE_SYSTEM_EXCLUSIVE_MESSAGE, data = Not used*

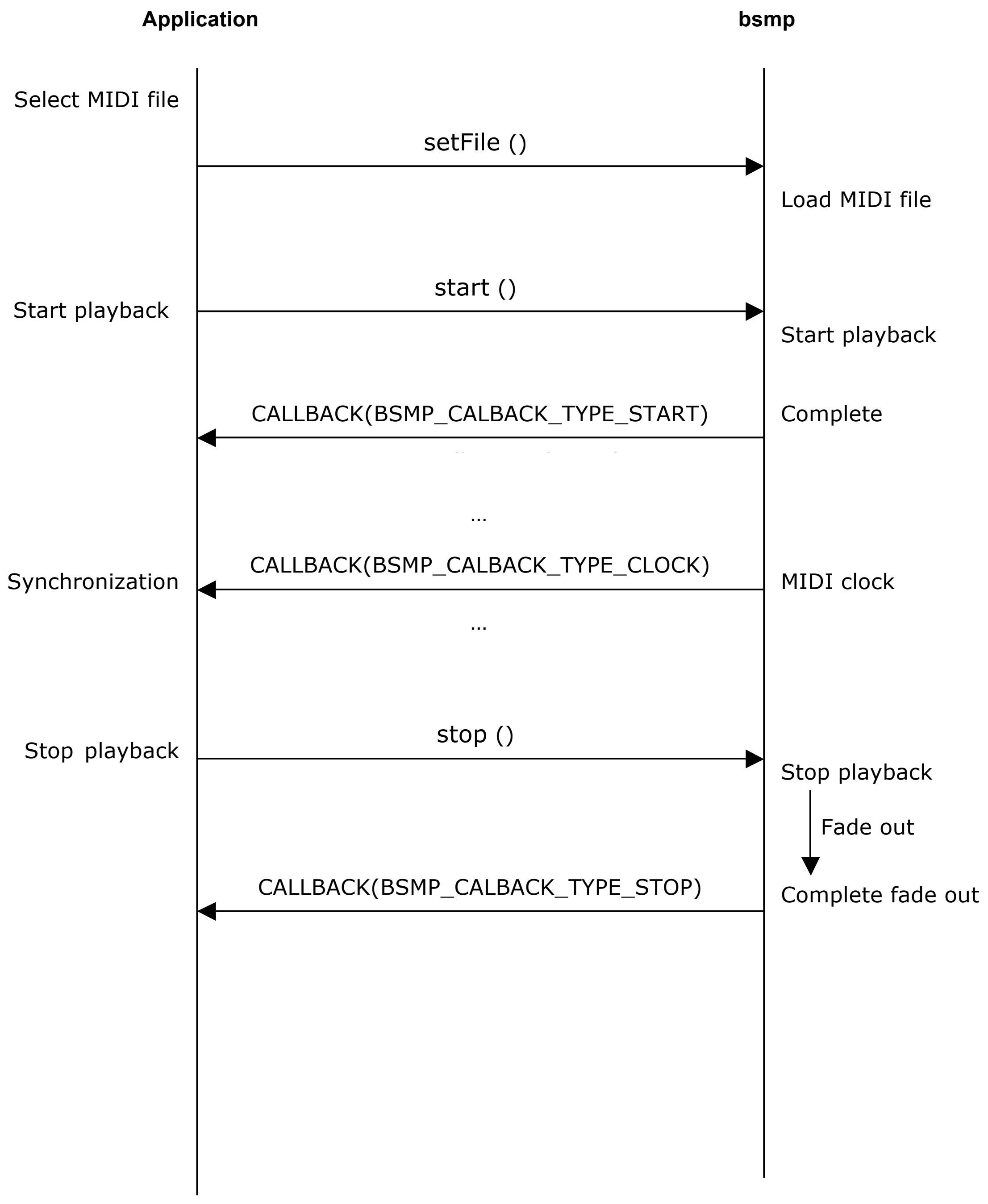
System exclusive message has been sent by player.

## 3.6. Sequences

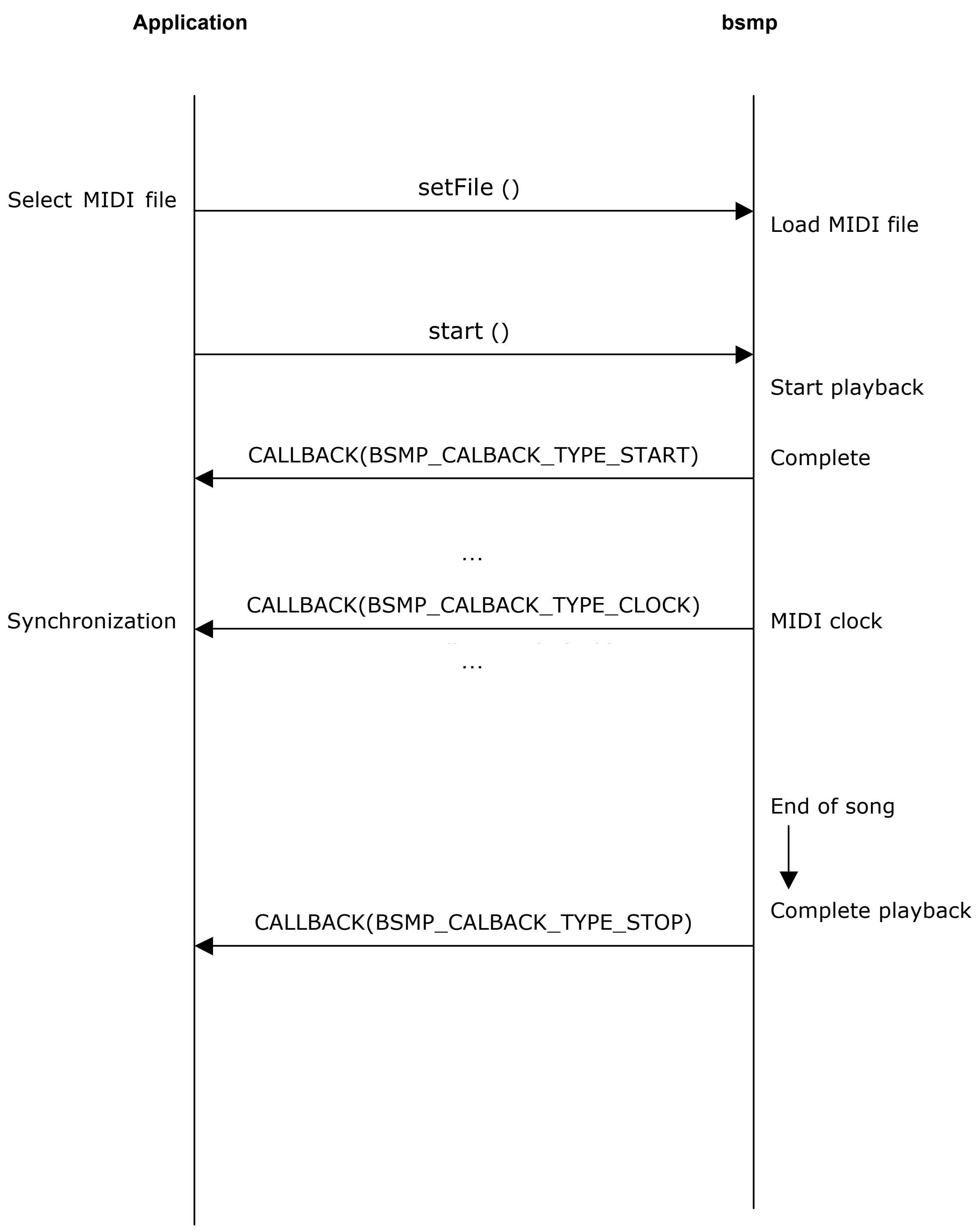
### 3.6.1. Initialization



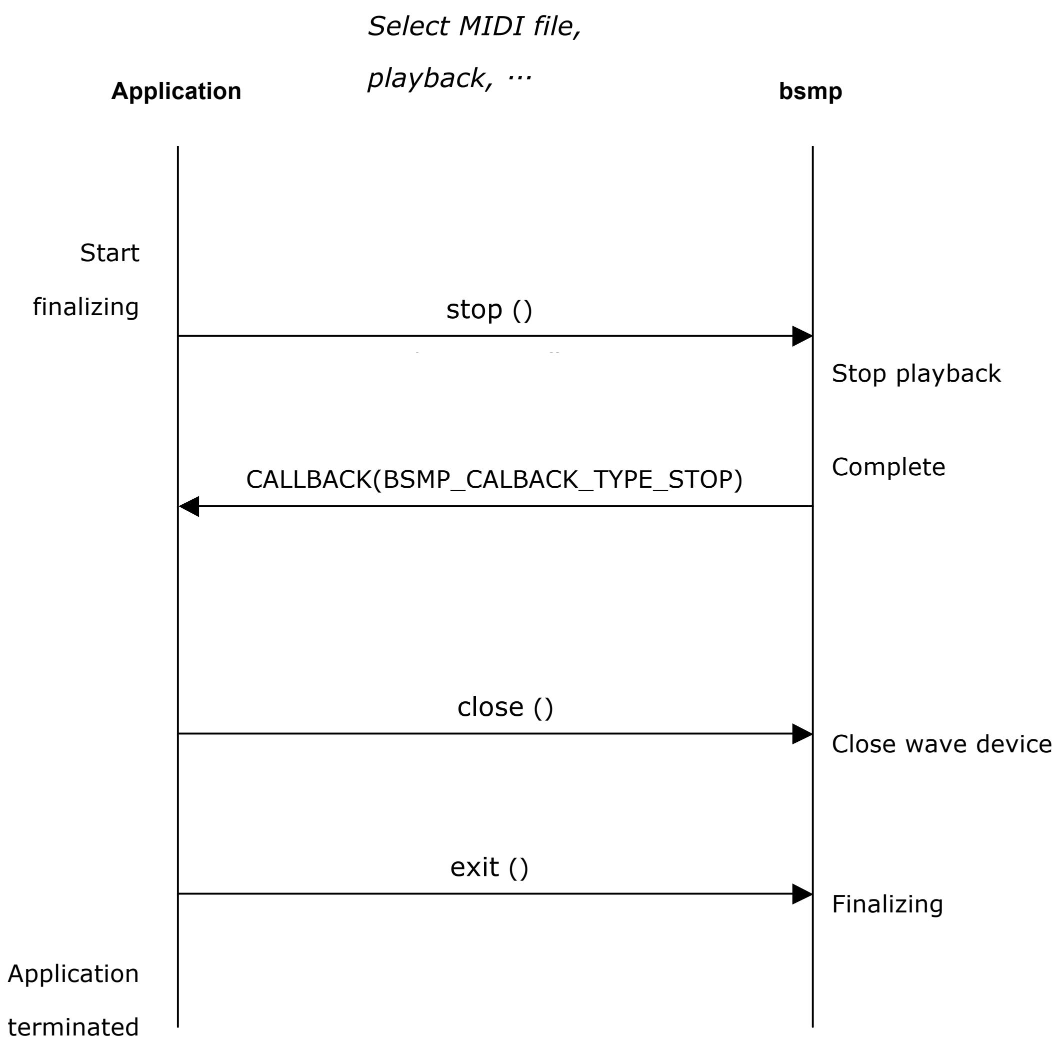
### 3.6.2. Specifying the MIDI Files - Start Playback – Stop by User



### 3.6.3. Specifying the MIDI File – Start Playback - End of the Song



## 3.6.4. Finalizing



# 4. MIDI Driver Library Specification

## 4.1. Constants

### 4.1.1. BSMD_ERR

typedef enum for result code.

| code | 内容 | |
|---|---|---|
| BSMD_OK | Success | |
| BSMD_ERR_PROTECTION | Protection error | |
| BSMD_ERR_INVALID_HANDLE | Invalid handle error | |
| BSMD_ERR_FILE | File error | |
| BSMD_ERR_MEMORY | Memory error | |
| BSMD_ERR_RESOURCE | Resource error | |
| BSMD_ERR_PARAM | Parameter error | |
| BSMD_ERR_AUDIO_DRIVER | Wave output error | |
| BSMD_ERR_DATA | Data error | |
| BSMD_ERR_MODULE | External module error | |
| BSMD_ERR_NOT_SUPPORTED | Unsupported error | |
| BSMD_ERR_UNDEFINED | Undefined | |

### 4.1.2. BSMD_CTRL

Typede enum for control API. Please refer to section 4.4.25 ctrl.

### 4.1.3. BSMD_CALLBACK_TYPE

Typedef enum for callback types. Please refer to section 4.5 Callback (BSMD_CALLBACK).

### 4.1.4. BSMD_SOUND_LIBRARY_SEL_MODE

typedef enum for selection modes of sound library files.

| code | 内容 | |
|---|---|---|
| BSMD_SOUND_LIBRARY_SEL_MODE_NORMAL | Default mode | |

## 4.2. Typedefs

### 4.2.1. BSMD_HANDLE

Handle for controlling this library.

### 4.2.2. BSMD_CALLBACK

Callback function type for sending information from this library to the user application. Please refer to section 4.5 Callback (BSMD_CALLBACK).

```
BSMD_CALLBACK ()
        Input:    BSMD_HANDLE handle            Effective handle of the library
                  BSMD_CALLBACK_TYPE type       Callback type
                  void *data                    Pointer of the data
                  void *user                    Pointer of the specified user area
        Output:   void
```

### 4.2.3. BSMD_LOAD

Function type for Geting the API table（BSMP_FUNC）.

## 4.3. Structures

### 4.3.1. BSMD_FUNC

Structure for API table. Please refer to section 4.4 API.

### 4.3.2. BSMD_SOUND_LIBRARY

Structure for specifying the sound library file.

```
typedef struct {
        int index; /* Index for the sound library file */
        LPCTSTR path; /* Full path of the sound library file */
} BSMD_SOUND_LIBRARY;
```

### 4.3.3. BSMD_SOUND_LIBRARY_MEMORY

Structure for specifying the sound library file mapped on the memory.

```
typedef struct {
        int index; /* Index for the sound library file */
        char *address; /* Memory address for the mapped sound library file */
        unsigned long *size; /* Size of the sound library file [Byte] */
} BSMD_SOUND_LIBRARY_MEMORY;
```

### 4.3.4. BSMD_SOUND_LIBRARY_SEL

Structure to specify relationship between each part and sound library files.

```
typedef struct {
        int module; /* Module index (0, 1, ...) */
        int part; /* Part index (0, 1, ..., 15) */
        int index; /* Index of the sound library file */
        BSMD_SOUND_LIBRARY_SEL_MODE mode; /* selection modes (section 4.1.4) */
} BSMD_SOUND_LIBRARY_SEL;
```

### 4.3.5. BSMD_FRAME

Structure for callback (BSMD_CALLBACK_TYPE_FRAME)

```
typedef struct {
        long sampleFrames; /* audio frame length [sample] */
        void *data; /* buffer for output audio (Signed 16bit, 2ch interleaved) */
} BSMD_FRAME;
```

## 4.4. API

### 4.4.1. initialize

```
BSMD_ERR initialize ()
    Input:
        BSMD_HANDLE *handle             Pointer of the handle (!= NULL)
        BSMD_CALLBACK callback          Pointer of the callback function
        void *user                      Pointer of the user area for callback
        void *target                    Target independent data
        const unsigned char *key        Key code
    Output:
        Error code
```

Initialize the library and Synthesizer Engine Library.

Synthesizer Engine Library loads the default sound library (from own resource, or from the defined path) to index #0.

Before using the library, the application have to call the on of initialize* () functions.

The application have to set 64 byte key code to the argument "key".

This functions requires the fixed processing time because of loading the sound library.

The application have to set the following values to argument "target"

- Win/WinCE: The handle of the parent window (HWND)
- Android: This library receives pointer of the following sturucture, and calls the Activity class method of your application using information this information.

```
typedef struct {
    JNIEnv *env;
    jobject thiz;
}
```

- Other OS: NULL

### 4.4.2. initializeWithSoundLib

```
BSMD_ERR initializeWithSoundLib ()
```

*Input:*

| | |
|---|---|
| *BSMD_HANDLE *handle* | *Pointer of the handle (!= NULL)* |
| *BSMD_CALLBACK callback* | *Pointer of the callback function* |
| *void *user* | *Pointer of the user area for callback* |
| *LPCTSTR libraryPath* | *Full path of the sound library file* |
| *void *target* | *Target independent data* |
| *const unsigned char *key* | *Key code* |

*Output:*

*Error code*

Initialize the library and Synthesizer Engine Library.

Synthesizer Engine Library loads the sound library file on the specified path to index #0.

### 4.4.3. initializeWithSoundLibMemory

*BSMD_ERR initializeWithSoundLibMemory ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE *handle* | *Pointer of the handle (!= NULL)* |
| *BSMD_CALLBACK callback* | *Pointer of the callback function* |
| *void *user* | *Pointer of the user area for callback* |
| *char *libraryAddress* | *Address of the mapped sound library* |
| *unsigned long librarySize* | *Size of the sound library file [Byte]* |
| *void *target* | *Target independent data* |
| *const unsigned char *key* | *Key code* |

*Output:*

*Error code*

Initialize the library and Synthesizer Engine Library.

Synthesizer Engine Library loads the sound library file on the specified memory to index #0.

### 4.4.4. exit

```
BSMD_ERR exit ()
        Input:
                BSMD_HANDLE handle        Effective handle of the library
        Output:
                Error code
```

Finalize the library.

The application have to call this function before termination. If the library is playing, the application have to stop playback before calling this function.

### 4.4.5. getNumDrivers

```
int getNumDrivers ()
        Input:
                BSMD_HANDLE handle        Effective handle of the library
        Output:
                The number of supported drivers.
```

Get the number of wave output drivers supported by the library.

### 4.4.6. getNumDevices

```
int getNumDevices ()
        Input:
                BSMD_HANDLE handle        Effective handle of the library
                LPCTSTR driver            Name of wave output driver
        Output:
                The number of available wave output devices
```

Get the number of available wave output devices in the specified wave output driver.

### 4.4.7. getDriverName

*LPCTSTR getDriverName ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |
| *int index* | *Index for the wave output driver* |

*Output:*

*Name of the specified wave output driver*

Get the name of the specified wave output driver.

### 4.4.8. getDeviceName

*LPCTSTR getDeviceName ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |
| *LPCTSTR driver* | *Name of the wave output driver* |
| *int index* | *Index for the wave output device* |

*Output:*

*Name of the specified wave output device*

Get the name of the specified wave output device.

### 4.4.9. showDeviceControlPanel

*void showDeviceControlPanel ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |
| *LPCTSTR driver* | *Name of the wave output driver* |
| *LPCTSTR device* | *Name of the wave output device* |

Display the control panes of the specified wave output device

### 4.4.10. open

*BSMD_ERR open ()*

*Input:*

```
                BSMD_HANDLE handle        Effective handle of the library
                LPCTSTR driver            Name of the wave output driver
                LPCTSTR device            Name of the wave output device
        Output:
                Error code
```

Open the specified wave output device. If the argument “driver” and “device” is NULL, default wave output driver and device will be selected automatically.

### 4.4.11. close

```
BSMD_ERR close ()
        Input:
                BSMD_HANDLE handle        Effective handle of the library
        Output:
                Error code
```

Close the wave output device.

### 4.4.12. start

```
BSMD_ERR start ()
        Input:
                BSMD_HANDLE handle        Effective handle of the library
        Output:
                Error code
```

Start Real-time MIDI function.

### 4.4.13. stop

```
BSMD_ERR stop ()
        Input:
                BSMD_HANDLE handle        Effective handle of the library
        Output:
```

```
                    Error code
```

Stop Real-time MIDI function.

### 4.4.14. isPlaying

```
int isPlaying ()
        Input:
                    BSMD_HANDLE handle          Effective handle of the library
        Output:
                    1: playing
                    0: not playing
```

Get the flag for the library’s Real-time function is enabled, or not.

### 4.4.15. setChannelMessage

```
void setChannelMessage ()
        Input:
                    BSMD_HANDLE handle          Effective handle of the library
                    unsigned char port          MIDI Port (0 = A, 1 = B, ...)
                    unsigned char status        MIDI Status (0x80□0xEF)
                    unsigned char data1         1st data (0x00□0x7F)
                    unsigned char data2         2nd data (0x00□0x7F)
```

Set MIDI channel message.

### 4.4.16. setSystemExclusiveMessage

```
void setSystemExclusiveMessage ()
        Input:
                    BSMD_HANDLE handle          Effective handle of the library
                    unsigned char port          MIDI Port (0 = A, 1 = B, ...)
                    unsigned char status        MIDI Status (0xF0)
                    unsigned char *data         Address of data array
```

| | |
|---|---|
| *int size* | *Length of data [byte]* |

Set MIDI system exclusive message.

### 4.4.17. setFile

| | | |
|---|---|---|
| *BSMD_ERR setFile ()* | | |
| *Input:* | | |
| | *BSMD_HANDLE handle* | *Effective handle of the library* |
| | *LPCTSTR path* | *Full path of the MIDI file* |
| *Output:* | | |
| | *Error code* | |

Specify the MIDI sequence file with file path. See **2.2 Inputs** for available file formats.

### 4.4.18. setFileMemory

| | | |
|---|---|---|
| *BSMD_ERR setFileMemory ()* | | |
| *Input:* | | |
| | *BSMD_HANDLE handle* | *Effective handle of the library* |
| | *char *address* | *Memory address for the mapped MIDI file* |
| | *long size* | *Size of the MIDI file [byte]* |
| *Output:* | | |
| | *Error code* | |

Specify the MIDI sequence file mapped on the memory controlled by the application.

### 4.4.19. getFileMemory

*BSMD_ERR getFileMemory ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |
| *char **address* | *Pointer of the memory address* |
| *long *size* | *Pointer of the file size [byte]* |

*Output:*

*Error code*

Get the memory address and size used for loading MIDI file. This memory is controlled by the library.

### 4.4.20. getFileInfo

*BSMD_ERR getFileInfo ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |
| *int *format* | *Pointer of the MIDI file format* |
| *unsigned short *division* | *Pointer of the MIDI file division [TPQN]* |
| *unsigned long *totaltick* | *Pointer of the number of tick* |
| *unsigned long *totaltime* | *Pointer of the length [s]* |

*Output:*

*Error code*

Get information of the specified MIDI sequence file.

### 4.4.21. startFilePlay

*BSMD_ERR startFilePlay ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |

*Output:*

*Error code*

Start playback of the specified MIDI file from current song position.

### 4.4.22. stopFilePlay

*BSMD_ERR stopFilePlay ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |

*Output:*

*Error code*

Stop playback of the specified MIDI file.

Calling this function means the application instructs the start of fade out process, and the playback still alive. The application has to detect the completion of the playback by the callback function described later.

Current song position will be saved after calling this function.

### 4.4.23. seekFilePlay

*BSMD_ERR seekFilePlay ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |
| *unsigned long tick* | *Song position [MIDI tick]* |

*Output:*

*Error code*

Specify song position.

### 4.4.24. isFilePlaying

*int isFilePlaying ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |

*Output:*

*1: playing*

*0: not playing*

Get the flag for the library is playing the MIDI file, or not.

### 4.4.25. ctrl

*BSMD_ERR ctrl ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |
| *BSMD_CTRL ctrl* | *Control target* |
| *void *data* | *Address of data* |
| *int size* | *Size of data [byte]* |

*Output:*

*Error code*

Do various operations.

| ctrl | data | | description |
|---|---|---|---|
| | type | I/O | |
| BSMD_CTRL_SET_SAMPLE_RATE | int | I | Set playback sample rate [Hz] |
| BSMD_CTRL_GET_SAMPLE_RATE | int | O | Get playback sample rate [Hz] |
| BSMD_CTRL_SET_CHANNELS | int | I | Not supported |
| BSMD_CTRL_GET_CHANNELS | int | O | Get number of output channels |
| BSMD_CTRL_SET_BLOCK_SIZE | long | I | Set frame size [sample] of wave output.<br><br>This value affects the latency of Real-time MIDI function.<br><br>In ASIO / AudioUnit drives, this value is overwrote by the device drivers. So the applications have to get this value after calling open in section 3.4.10, using BSMD_CTRL_GET_BLOCK_SIZE. |
| BSMD_CTRL_GET_BLOCK_SIZE | long | O | Get frame size [sample] of wave output. |
| BSMD_CTRL_SET_BUFFERS | int | I | Set number of frames for wave output.<br><br>This value affects the latency of Real-time MIDI function.<br><br>In ASIO / AudioUnit drivers, this value is fixed (= 1). |
| BSMD_CTRL_GET_BUFFERS | int | O | Get number of frames for wave output. |
| BSMD_CTRL_SET_POLY | int | I | Set polyphonic number of synthesizer |
| BSMD_CTRL_GET_POLY | int | O | Get polyphonic number of synthesizer |

| ctrl | data | | description |
|---|---|---|---|
| | type | I/O | |
| BSMD_CTRL_SET_MASTER_VOLUME | int | I | Set playback volume (BSMP_VOLUME_MIN ~ BSMP_VOLUME_MAX). The default value is BSMP_VOLUME_DEF. |
| BSMD_CTRL_GET_MASTER_VOLUME | int | O | Get playback volume |
| BSMD_CTRL_SET_MASTER_KEY | int | I | Set playback key (BSMD_KEY_MIN ~ BSMD_KEY_MAX). The unit of the values is 100[cent], and the default value is BSMD_KEY_DEF. This value is not cleared on the end of the playback. |
| BSMD_CTRL_GET_MASTER_KEY | int | O | Get playback key. |
| BSMD_CTRL_SET_MASTER_TUNE | int | I | Set fine tuning (BSMD_TUNE_MIN ~ BSMD_TUNE_MAX). The unit of the values is 1[cent], and the default value is BSMD_TUNE_DEF. This value is not cleared on the end of the playback. |
| BSMD_CTRL_GET_MASTER_TUNE | int | O | Get fint tuning. |
| BSMD_CTRL_SET_SPEED | int | I | Set playback speed. (BSMD_SPEED_MIN ~ BSMD_SPEED_MAX). The unit of the value is 1[%], and the default value is BSMD_SPEED_DEF. This value is not cleared on the end of the playback. |
| BSMD_CTRL_GET_SPEED | int | O | Get playback speed. |

| ctrl | data | | description |
|---|---|---|---|
| | Type | I/O | |
| BSMD_CTRL_SET_REVERB | *int* | I | Set effectiveness of reverb. This value is not cleared on the end of the playback. |
| BSMD_CTRL_GET_REVERB | *int* | O | Get effectiveness of reverb |
| BSMD_CTRL_GET_REVERB _AVAILABLE | *int* | O | Get availability of reverb |
| BSMD_CTRL_SET_CHORUS | *int* | I | Set effectiveness of chorus. This value is not cleared on the end of the playback. |
| BSMD_CTRL_GET_CHORUS | *int* | O | Get effectiveness of chorus |
| BSMD_CTRL_GET_CHORUS _AVAILABLE | *int* | O | Get availability of chorus |
| BSMD_CTRL_SET_DELAY | *int* | I | Set effectiveness of delay. This value is not cleared on the end of the playback. |
| BSMD_CTRL_GET_DELAY | *int* | O | Get effectiveness of delay |
| BSMD_CTRL_GET_DELAY _AVAILABLE | *int* | O | Get availability of delay |
| BSMD_CTRL_SET_REVERB_HQ | *int* | I | Set HQ Reverb (1: On, 0: Off, Customized version only) |

| ctrl | data | | description |
|---|---|---|---|
| | type | I/O | |
| BSMD_CTRL_GET_SOUND_LIBRARY_NUM | int | O | Get number of the slots for sound libraries |
| BSMD_CTRL_SET_SOUND_LIBRARY | BSMD_SOUND_LIBRARY | I | Set sound library with file path |
| BSMD_CTRL_SET_SOUND_LIBRARY_MEMORY | BSMD_SOUND_LIBRARY_MEMORY | I | Set sound library with memory |
| BSMD_CTRL_SET_SOUND_LIBRARY_SEL | BSMD_SOUND_LIBRARY_SEL | I | Set selection mode for the loaded sound library |
| BSMD_CTRL_GET_SOUND_LIBRARY_SEL | BSMD_SOUND_LIBRARY_SEL | I/O | Get selection mode for the loaded sound library |
| BSMD_CTRL_SET_NUMBER_OF_REGIONS | int | I | Set maximum number of region in each instrument |

| ctrl | data | | description |
|---|---|---|---|
| | type | I/O | |
| BSMD_CTRL_GET_INSTRUMENT_NAME ~ BSMD_CTRL_GET_INSTRUMENT_NAME + 15 | char (TCHAR) | O | Get instrument name of the specified part (Ch1~16) |
| BSMD_CTRL_SET_MUTE ~ BSMD_CTRL_SET_MUTE + 15 | int | I | Set mute (0: Off, 1: On) to the specified part (Ch1~16) |
| BSMD_CTRL_GET_MUTE ~ BSMD_CTRL_GET_MUTE + 15 | int | O | Get mute (0: Off, 1: On) of the specified part (Ch1~16) |
| BSMD_CTRL_SET_SOLO ~ BSMD_CTRL_SET_SOLO + 15 | int | I | Set solo (0: Off, 1: On) to the specified part (Ch1~16) |
| BSMD_CTRL_GET_SOLO ~ BSMD_CTRL_GET_SOLO + 15 | int | O | Get solo (0: Off, 1: On) of the specified part (Ch1~16) |

| *ctrl* | data | | description |
|---|---|---|---|
| | type | I/O | |
| *BSMD_CTRL_GET_AUDIO_UNIT* | | | Get AudioUnit |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

### 4.4.26. version

*void version ()*

*Input:*

| | |
|---|---|
| *BSMD_HANDLE handle* | *Effective handle of the library* |
| *LPTSTR engine* | *Version of Synthesizer Engine Library* |
| *int engineSize* | *Length of engine* |
| *LPTSTR driver* | *Version of MIDI Driver Library* |
| *int driverSize* | *Length of driver* |

*Output:*

*void*

Get the name of MIDI Driver Library and Synthesizer Engine Library.

## 4.5. Callback (BSMD_CALLBACK)

Callback function provides various information to the application. It is specified on 4.4.1 initialize, with function type defined in section 4.2.2. BSMD_CALLBACK.
Each callback is called from calculation thread of synthesizer. So the application can not spend long duration on receiving them.

### 4.5.1. Open

*type = BSMD_CALLBACK_TYPE_OPEN, data = Not used*

Wave output driver has been opened

### 4.5.2. Close

*type = BSMD_CALLBACK_TYPE_CLOSE, data = Not used*

Wave output driver has been closed

### 4.5.3. Start

*type = BSMD_CALLBACK_TYPE_START, data = Not used*

Real-time MIDI function has been started

### 4.5.4. Stop

*type = BSMD_CALLBACK_TYPE_STOP, data = Not used*

Real-time MIDI function has been stopped

### 4.5.5. Audio Frame

*type = BSMD_CALLBACK_TYPE_FRAME, data = (BSMD_FRAME *) frameData*

Called on every frames of wave output process

### 4.5.6. File Start

*type = BSMD_CALLBACK_TYPE_FILE_START, data = Not used*

Playback has been started

### 4.5.7. File Stop

*type = BSMD_CALLBACK_TYPE_FILE_STOP, data = (unsigned long *) errorcode*

Playback has been stopped

errorcode:

*0□Normal*

*BSMD_ERR_AUDIO_DRIVER□Error stop by wave output driver*

*BSMD_ERR_DATA□Error stop by data*

### 4.5.8. File Seek

*type = BSMP_CALLBACK_TYPE_FILE_SEEK, data = □□□*

Playback song position has been changed.

If your application calculates song position using 4.5.9 MIDI Clockcallback, please reset song position to start, tempo to 120[BPM], on receiving this callback.

### 4.5.9. MIDI Clock

*type = BSMP_CALLBACK_TYPE_CLOCK, data = □□□*

Standard MIDI clock (24[TPQN])

### 4.5.10. Tempo

*type = BSMP_CALLBACK_TYPE_TEMPO, data = (unsigned long *) tempo*

Playback tempo has been changed ([usec/beat])

### 4.5.11. Time Signature

*type = BSMP_CALLBACK_TYPE_TIME_SIGNATURE, data = (unsigned long *) timeSignature*

Playback time signature（nn/dd/cc/bb）has been changed.

### 4.5.12. Channel Message

*type = BSMP_CALLBACK_TYPE_CHANNEL_MESSAGE, data = (unsigned long *) data*

Channel message has been sent by player

bit 31-24: MIDI Port (0x00 ~ )

bit 23 - 16: Status byte (0x90 ~ 0xEF)

bit 15 – 8 : First Data (0x00 ~ 0x7F)

bit 7 – 0 : Second Data (0x00 ~ 0x7F)

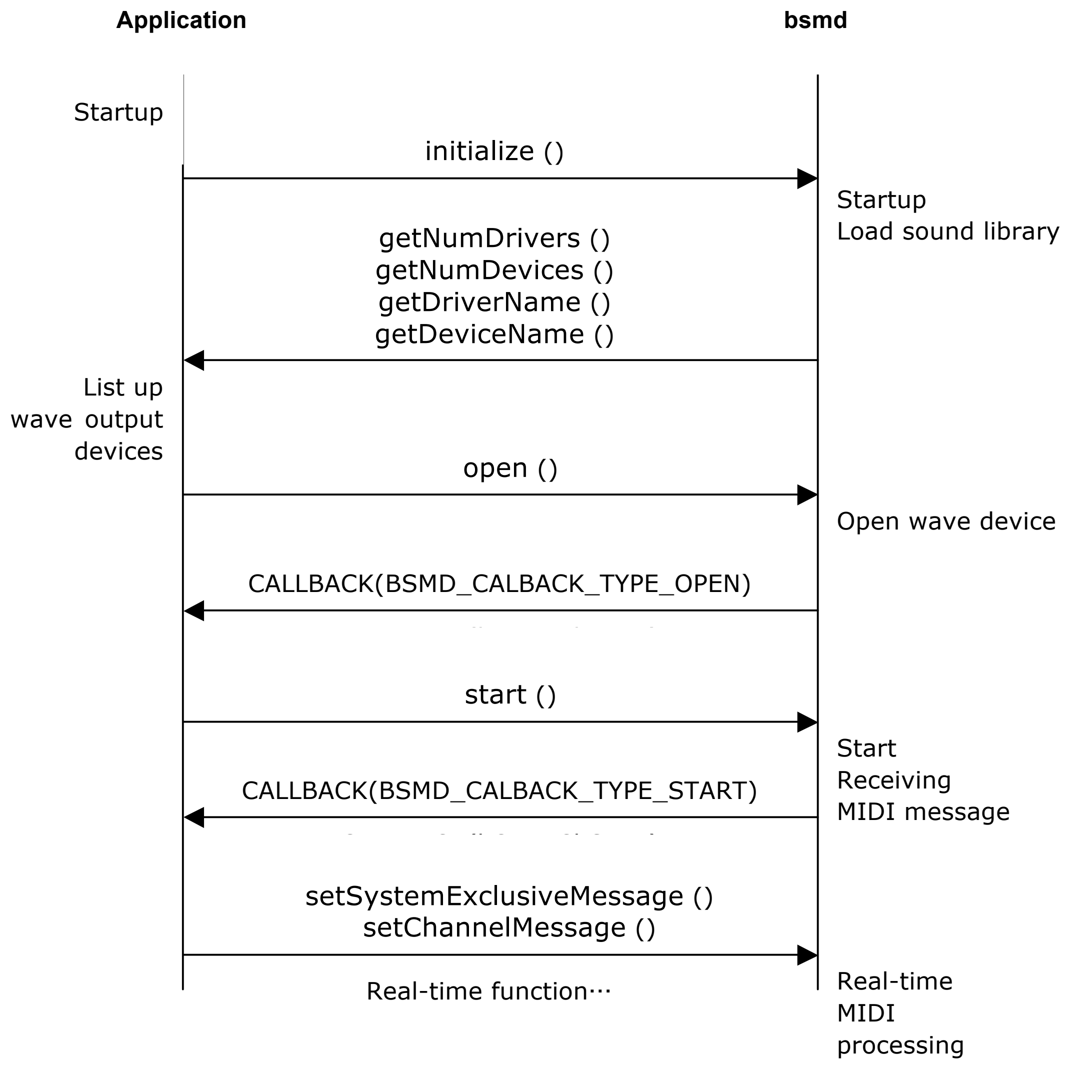
### 4.5.13. System Exclusive Message

*type = BSMP_CALLBACK_TYPE_SYSTEM_EXCLUSIVE_MESSAGE, data = Not used*

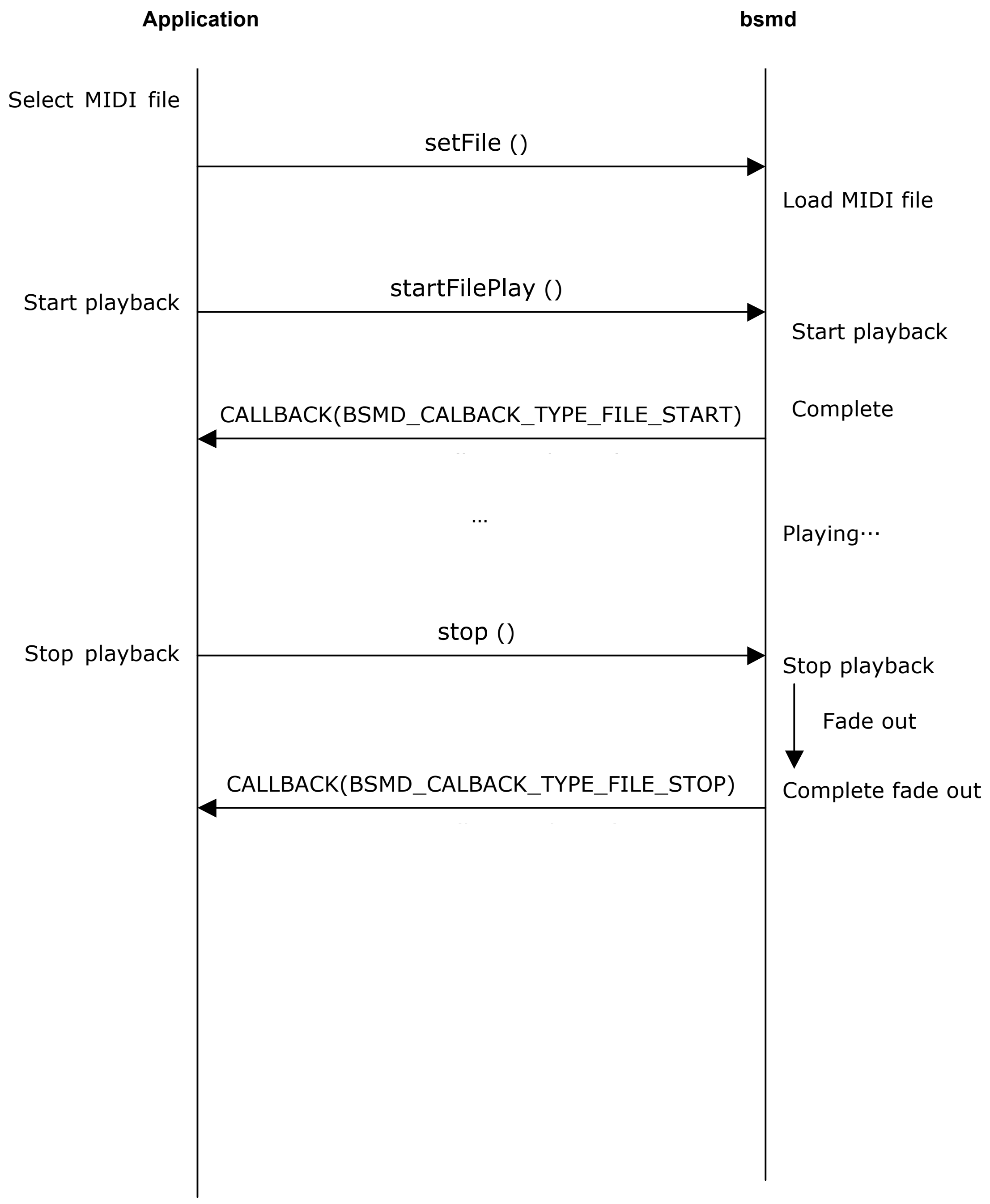
System exclusive message has been sent by player.

## 4.6. Sequences

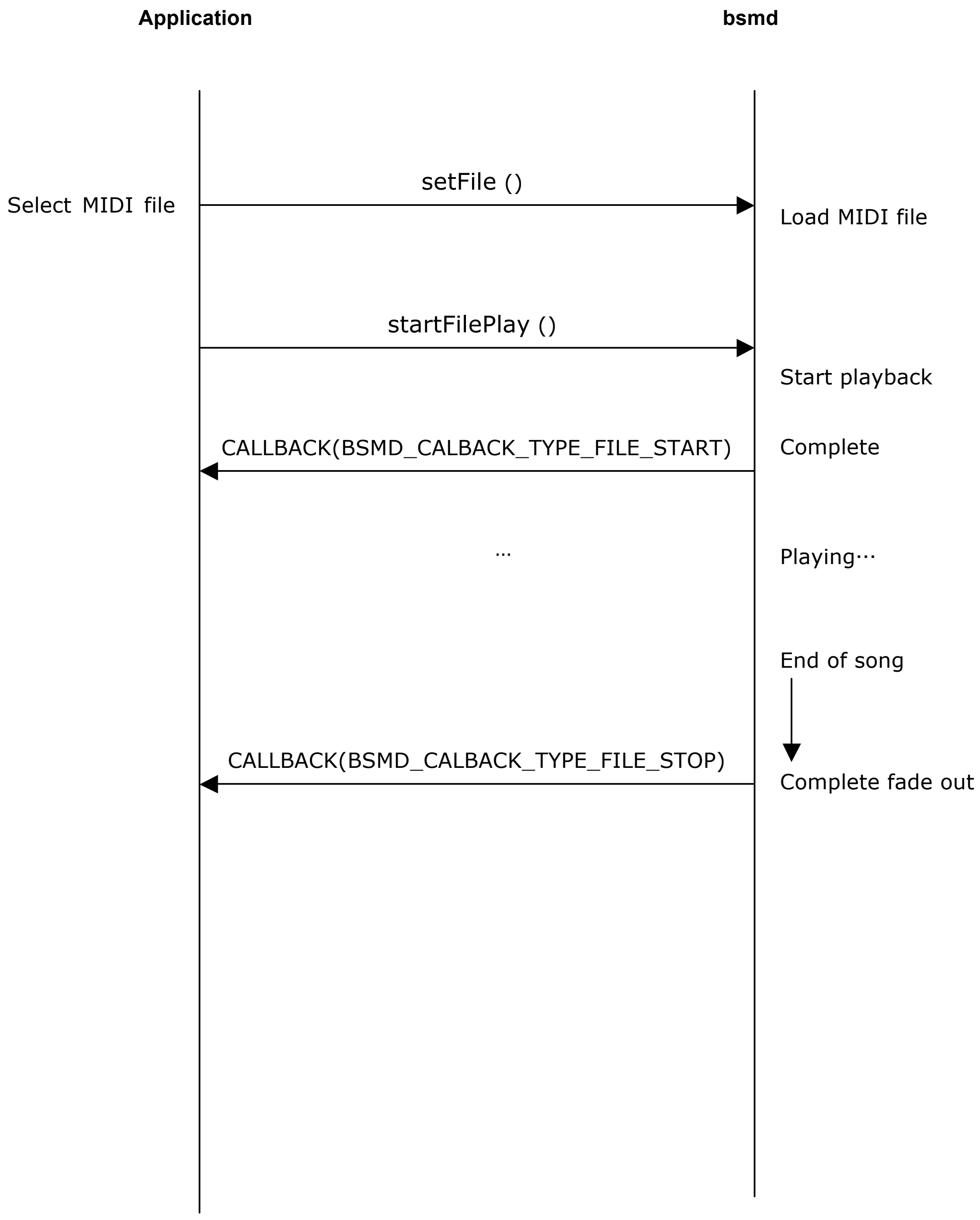
### 4.6.1. Initializing



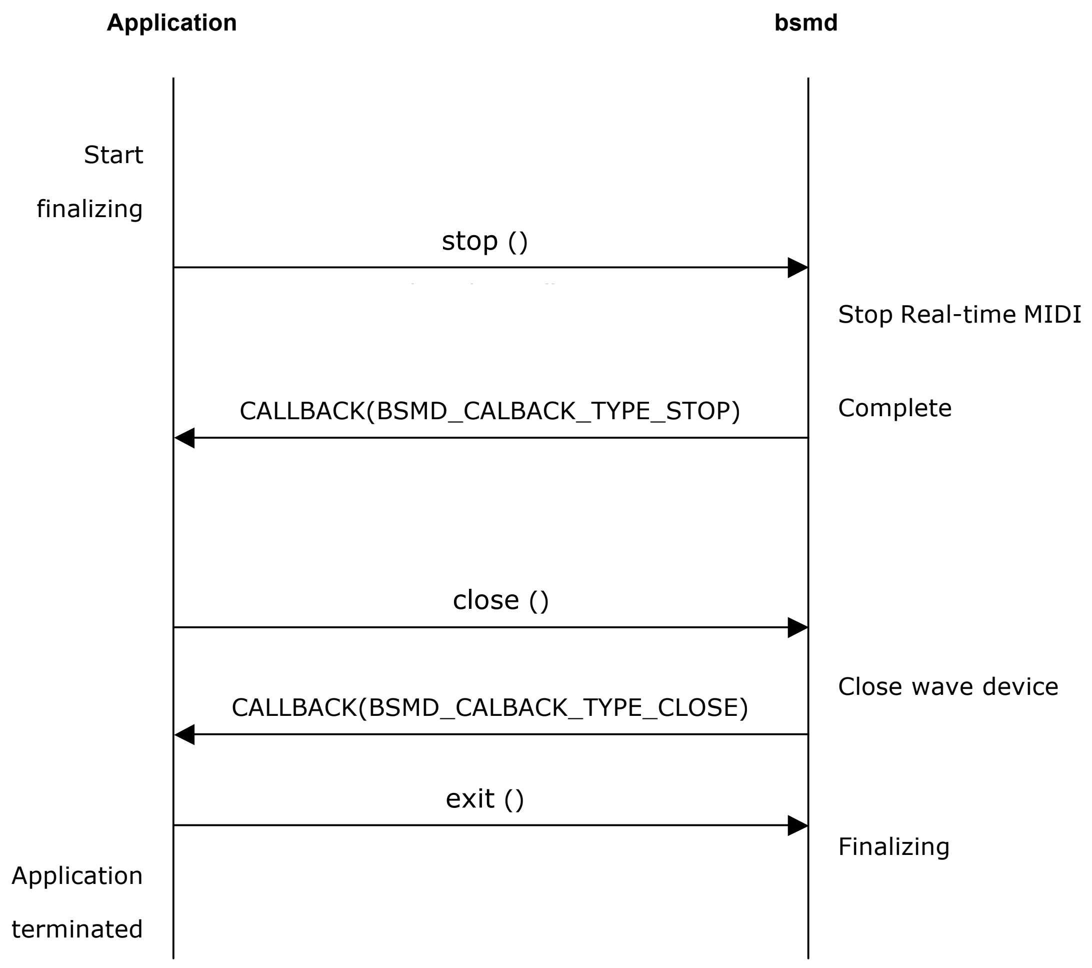
### 4.6.2. Specifying the MIDI Files – Start Playback – Stop by User



### 4.6.3. Specifying the MIDI File – Start Playback – End of the Song



### 4.6.4. Finalizing



## 5. Appendix

### 5.1. About DLS File Format

Wave format in <wave-list> chunk should satisfy following specification.

- linear PCM
- monaural

Following modulation routings are not supported. All parameters work with default value.

- Key Number Generator
  - MIDI Note to Key
  - RPN2 to Key
- Filter
  - Mod LFO CC1 to Fc
  - Mod LFO Channel Press. to Fc
- Gain
  - Mod LFO CC1 to Gain
  - Mod LFO Chan. Press. to Gain
  - Velocity to Gain
  - MIDI CC7 to Gain
  - MIDI CC11 to Gain
- Pitch
  - Pitch Wheel RPN0 to Pitch
  - RPN1 to Pitch
  - Vib LFO CC1 to Pitch
  - Vib LFO Chan. Press. to Pitch
  - Mod LFO CC1 to Pitch
  - Mod LFO Chan. Press. to Pitch
- Output
  - MIDI CC10 to Pan
  - Default Reverb Send
  - Default Chorus Send