MIDI Clip File Specification

Standard MIDI File
Using Universal MIDI Packet for MIDI 1.0 and MIDI 2.0 Protocols

MIDI Association Document: M2-116-U

Document Version 1.0
Draft Date May 11, 2023

Published June 15, 2023

Developed and Published By
The MIDI Association
and
Association of Musical Electronics Industry (AMEI)

PREFACE

MIDI Association Document M2-116-U
MIDI Clip File Specification

This MIDI Clip File Specification is a file format for MIDI sequences using the Universal MIDI Packet data format. MIDI 2.0 introduced the Universal MIDI Packet, a data format which supports all the original MIDI 1.0 Protocol messages as well as all MIDI 2.0 Protocol messages. The MIDI Clip File serves a role which is similar to the version 1 Standard MIDI File Type 0, with all data in a single sequence of Universal MIDI Packet messages. This MIDI Clip File Specification has a complementary specification, the MIDI Container File specification which is similar to the version 1 Standard MIDI File Type 1. The MIDI Container File supports the Universal MIDI Packet data format by containing one or more MIDI Clip Files, arranged in one or more tracks.

http://www.amei.or.jp

https://www.midi.org

Version History

Table 1 Version History

Publication Date	Version	Changes
June 15, 2023	1.0	Initial release

Contents

Figures

Tables

1 Introduction

1.1 Executive Summary

This MIDI Clip File specification defines a Standard MIDI File format which supports all MIDI 1.0 data and all MIDI 2.0 data.

Standard MIDI Files (SMF) have been an important part of the MIDI ecosystem since 1988. A Standard MIDI File is an interchange file format for saving MIDI sequences and opening them with other programs.

> *For example, a composer might save a Standard MIDI File created by a notation program and open it in a MIDI sequencing program, which will understand the MIDI events such as notes and controllers as well as various parameters of the file, such as track names, tempo changes, etc.*

There are also established markets for Standard MIDI Files that include:

- Publishing (scores with SMF files included)
- Music education
- Karaoke
- MIDI backing tracks
- Standard MIDI Files that support General MIDI to allow consistent playback
- Web page authoring

MIDI 2.0 adds new capabilities to preserve and grow these market applications.

The original Standard MIDI File formats cannot support MIDI 2.0 data. This MIDI Clip File specification defines the use of the Universal MIDI Packet (UMP) data format to support all MIDI 1.0 data and all MIDI 2.0 data. It serves a role which is similar to the Standard MIDI File Type 0 in MIDI 1.0, with all data in a single sequence of Universal MIDI Packet messages.

This MIDI Clip File Specification has a complementary specification, the MIDI Container File Specification which is similar to the Standard MIDI File Type 1 in MIDI 1.0. This MIDI Container File supports the Universal MIDI Packet data format by containing one or more MIDI Clip Files, arranged in one or more tracks. This container format also expands the applications and markets for Standard MIDI Files by allowing the inclusion of other media files including musical notation, audio, video, and application-specific files.

The MIDI Clip File also prompted the definition of many new MIDI messages in the UMP Format and a new UMP Message Type.

The Standard MIDI File version 1 specifications for Type 0 and Type 1 defined certain non-MIDI data as Meta Events. In fact, MIDI SMF1 was developed before the concept of metadata was well defined in the multimedia industry and was ahead of its time in codifying authors, publishers, tempo, key signatures and other aspects of musical performance that have now become standardized for the Internet.

MIDI 2.0 provides more space to define new messages, so these Meta Events have been replaced by new MIDI messages that can be sent over MIDI transports. These new messages can be found in the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

1.2 Background

Standard MIDI File types were defined in MIDI 1.0 as interchange files, commonly supported file formats to exchange MIDI data between different sequencers. MIDI 2.0 introduces a new data format, the Universal MIDI Packet, which cannot be represented in the previous Standard MIDI File formats. Therefore, new file formats are required.

The original Standard MIDI Specification defines three file formats:

1. **Type 0:** These files consist of one track, with note and other events being tagged with the MIDI channels to which they belong. These are often used to stream directly from a storage device (like a USB memory stick) and be played directly on a MIDI device with a tone generator.
2. **Type 1:** A sequence is saved as separate tracks, even if more than one track is assigned to the same MIDI channel. Track labels are also maintained. These types of SMF are often used for data exchange between different MIDI applications.
3. **Type 2:** Same as Type 1 (separate tracks), but each track may have its own tempo. Type 2 files have been rarely supported.

Two additional file formats are defined to support MIDI 2.0:

1. **MIDI Clip File:** The MIDI Clip File is a file format which supports the Universal MIDI Packet data format. The Universal MIDI Packet supports all the original MIDI 1.0 Protocol messages as well as all the newly defined MIDI 2.0 Protocol messages. The file format serves a role which is similar to the Standard MIDI File Type 0, with all data in a single sequence of Universal MIDI Packet messages.

   All data which were previously defined as Meta Events in version 1 Standard MIDI File are now defined as new MIDI messages in the Universal MIDI Packet Format. These new messages can be found in the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.
2. **MIDI Container File:** The MIDI Clip File has a complementary specification, the MIDI Container File format which is similar to the Standard MIDI File Type 1. This file format supports the Universal MIDI Packet data format by containing one or more MIDI Clip Files, arranged in one or more tracks. The format also supports the inclusion of other media files including musical notation, audio, video, and application-specific files.

1.3 References

1.3.1 Normative References

[MA01] ***Complete MIDI 1.0 Detailed Specification***, Document Version 96.1, Third Edition, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA02] ***M2-100-U MIDI 2.0 Specification Overview)***, Version 1.1, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA03] ***M2-101-UM MIDI Capability Inquiry (MIDI-CI)***, Version 1.2, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA04] ***M2-102-U Common Rules for MIDI-CI Profiles***, Version 1.1, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA05] ***M2-103-UM Common Rules for Property Exchange***, Version 1.1, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

[MA06] ***M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol***, Version 1.1, Association of Musical Electronics Industry, http://www.amei.or.jp/, and The MIDI Association, https://www.midi.org/

1.4 Terminology

1.4.1 Definitions

AMEI: Association of Musical Electronics Industry. Authority for MIDI Specifications in Japan.

Clock: An expression of musical progression, as measured in bars and beats (and further subdivisions).

Device: An entity, whether hardware or software, which can send and/or receive MIDI messages.

Group: A field in the UMP Format addressing some UMP Format MIDI messages (and some UMPs comprising a MIDI message) to one of 16 Groups. See the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

MIDI 1.0 Protocol: Version 1.0 of the MIDI Protocol as originally specified in ***[MA01]*** and extended by MA and AMEI with numerous additional MIDI message definitions and Recommended Practices. The native format for the MIDI 1.0 Protocol is a byte stream, but it has been adapted for many different transports. MIDI 1.0 messages can be carried in UMP packets. The UMP format for the MIDI 1.0 Protocol is defined in the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

MIDI 1.0 Specification: Complete MIDI 1.0 Detailed Specification, Document Version 96.1, Third Edition ***[MA01]***.

MIDI 2.0: The MIDI environment that encompasses all of MIDI 1.0, MIDI-CI, Universal MIDI Packet (UMP), MIDI 2.0 Protocol, MIDI 2.0 messages, and other extensions to MIDI as described in AMEI and MA specifications.

MIDI 2.0 Protocol: Version 2.0 of the MIDI Protocol. The native format for MIDI 2.0 Protocol messages is UMP as defined in M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

MIDI-CI: MIDI Capability Inquiry ***[MA03]***, a specification published by The MIDI Association and AMEI.

MA: MIDI Association.

MIDI Association: Authority for MIDI specifications worldwide except Japan. See also MMA.

MIDI Manufacturers Association: A California nonprofit 501(c)6 trade organization, and the legal entity name of the MIDI Association.

MMA: MIDI Manufacturers Association.

MUID (MIDI Unique Identifier): A 28-bit random number generated by a Device used to uniquely identify the Device in MIDI-CI messages sent to or from that Device.

PE: Property Exchange.

Profile: An MA/AMEI specification that includes a set of MIDI messages and defined responses to those messages. A Profile is controlled by MIDI-CI Profile Negotiation Transactions. A Profile may have a defined minimum set of mandatory messages and features, along with some optional or recommended messages and features. See the MIDI-CI specification ***[MA03]*** and the Common Rules for MIDI-CI Profiles ***[MA04]***.

Property Exchange: A set of MIDI-CI Transactions by which one device may access Property Data from another device.

Protocol: There are two defined MIDI Protocols: the MIDI 1.0 Protocol and the MIDI 2.0 Protocol, each with a data structure that defines the semantics for MIDI messages. See ***[MA01]*** and ***[MA06]***.

Receiver: A MIDI Device which has a MIDI Transport connected to its MIDI In. A MIDI Device is not required to recognize or act upon any specific MIDI messages that it receives in order to be defined as a Receiver.

Resource: A defined collection of one or more PE Properties with an associated inquiry to access its Properties.

Sender: A MIDI Device which transmits MIDI messages to a MIDI Transport which is connected to its MIDI Out or to its MIDI Thru port.

Tempo: The rate at which a passage of music is or should be played, declared as and measured in a number of Clocks per a unit of Time (typically beats per minute).

Transaction: An exchange of MIDI messages between two MIDI Devices with a bidirectional connection. All the MIDI messages in a single Transaction are associated and work together to accomplish one function. The simplest Transaction generally consists of an inquiry sent by one MIDI Device and an associated reply returned by a second MIDI Device. A Transaction may also consist of an inquiry from one MIDI Device and several associated replies from a second MIDI Device. A Transaction may be a more complex set of message exchanges, started by an initial inquiry from one MIDI Device and multiple, associated replies exchanged between the first MIDI Device and a second MIDI Device. Also see MIDI-CI Transaction.

UMP: Universal MIDI Packet, see ***[MA06]***.

UMP Format: Data format for fields and messages in the Universal MIDI Packet, see ***[MA06]***.

UMP Message: A MIDI message in the Universal MIDI Packet Format, see ***[MA06]***.

Universal MIDI Packet (UMP): The Universal MIDI Packet is a data container which defines the data format for all MIDI 1.0 Protocol messages and all MIDI 2.0 Protocol messages. UMP is intended to be universally applicable, i.e., technically suitable for use in any transport where MA/AMEI elects to officially support UMP. For detailed definition see M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

1.4.2 Reserved Words and Specification Conformance

In this document, the following words are used solely to distinguish what is required to conform to this specification, what is recommended but not required for conformance, and what is permitted but not required for conformance:

Table 2 Words Relating to Specification Conformance

Word	Reserved For	Relation to Specification Conformance
shall	Statements of requirement	**Mandatory** A conformant implementation conforms to all 'shall' statements.
should	Statements of recommendation	**Recommended but not mandatory** An implementation that does not conform to some or all 'should' statements is still conformant, providing all 'shall' statements are conformed to.
may	Statements of permission	**Optional** An implementation that does not conform to some or all 'may' statements is still conformant, providing that all 'shall' statements are conformed to.

By contrast, in this document, the following words are never used for specification conformance statements; they are used solely for descriptive and explanatory purposes:

Table 3 Words Not Relating to Specification Conformance

Word	Reserved For	Relation to Specification Conformance
must	Statements of unavoidability	Describes an action to be taken that, while not required (or at least not directly required) by this specification, is unavoidable. Not used for statements of conformance requirement (see 'shall' above).
will	Statements of fact	Describes a condition that as a question of fact is necessarily going to be true, or an action that as a question of fact is necessarily going to occur, but not as a requirement (or at least not as a direct requirement) of this specification. Not used for statements of conformance requirements (see 'shall' above).
can	Statements of capability	Describes a condition or action that a system element is capable of possessing or taking. Not used for statements of conformance permission (see 'may' above).
might	Statements of possibility	Describes a condition or action that a system element is capable of electing to possess or take. Not used for statements of conformance permission (see 'may' above).

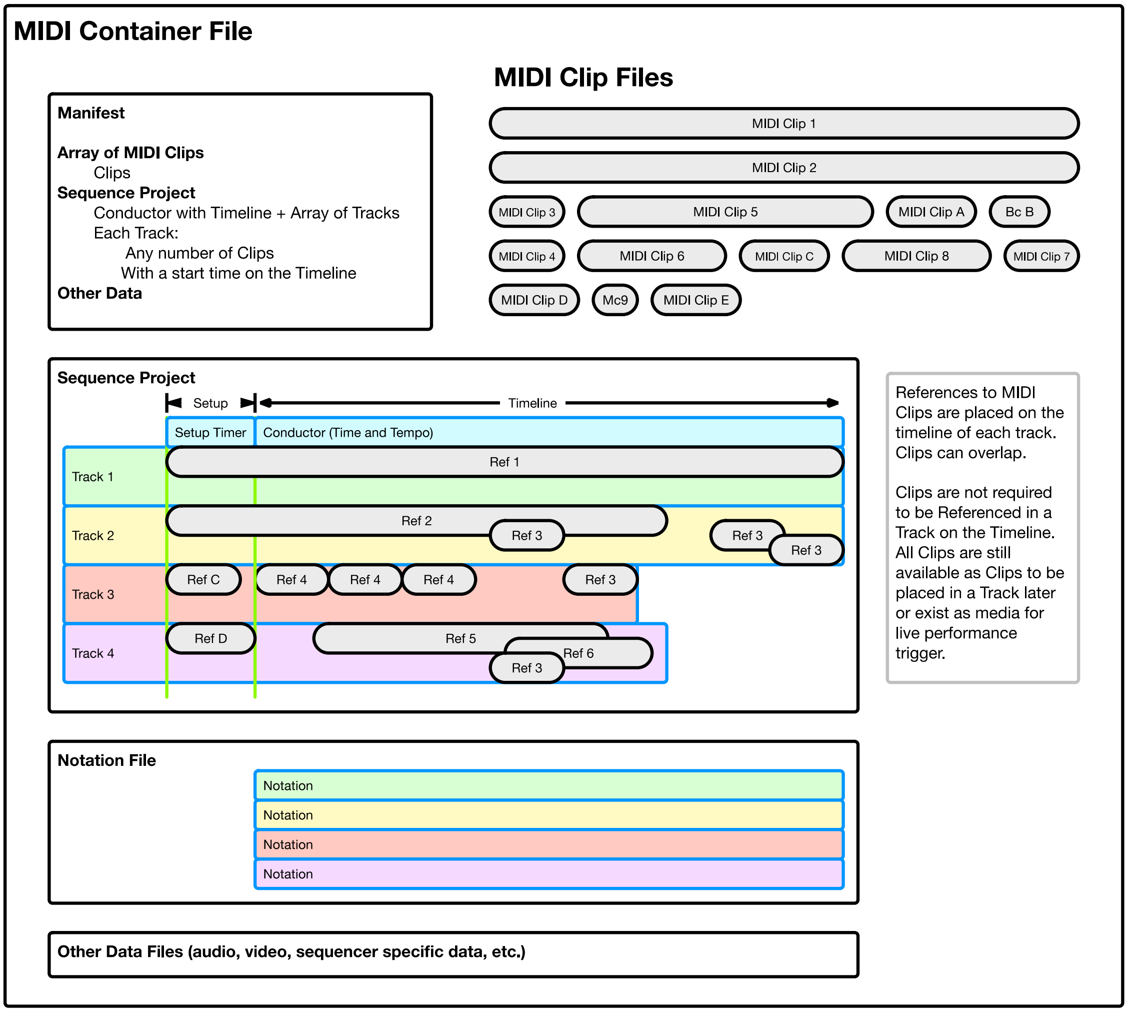
2 MIDI Clip File and Other Standard MIDI Files

The MIDI Clip File is a file format for MIDI data in a single sequence of Universal MIDI Packet messages.

MIDI Clip Files fill roles which are similar to the Type 0 file format defined in the version 1 Standard MIDI File specification but has several technical differences:

1. MIDI Clip Files use the Universal MIDI Packet Format as the native data format to support all MIDI 1.0 Protocol Messages and all MIDI 2.0 Protocol Messages.
2. All data which were previously defined as "Meta Events" in version 1 Standard MIDI File are now defined as new MIDI messages in the Universal MIDI Packet Format.
3. The MIDI Clip File design does not use the chunk concept in version 1 Standard MIDI Files.
4. MIDI Clip Files have a Clip Configuration Header to contain configuration data which may be used to set up a Receiver before the sequence begins to play.

A MIDI Clip File can be a stand-alone file or be a component inside a MIDI Container File which fills roles similar to the version 1 Standard MIDI File Type 1. A MIDI Container File supports the Universal MIDI Packet data format by containing MIDI Clip Files available for use in one or more tracks.



References to MIDI Clips are placed on the timeline of each track. Clips can overlap.

Clips are not required to be Referenced in a Track on the Timeline. All Clips are still available as Clips to be placed in a Track later or exist as media for live performance trigger.

Figure 1 MIDI Clip Files in a MIDI Container File

3 Data Format

The UMP Format is the data format for the payload of a MIDI Clip File. MIDI 1.0 Protocol messages and MIDI 2.0 Protocol messages in a MIDI Clip File are expressed in the UMP Format. A Delta Clockstamp message precedes every UMP to define the timing of each message.

Only the 8-byte File Header is not expressed as a UMP message. See Section ***5***.

All data in a MIDI Clip File is stored big endian.

3.1 Universal MIDI Packet (UMP)

The UMP Format is a data format for MIDI 1.0 Protocol and MIDI 2.0 Protocol messages. The UMP Format is defined in the M2-104-UM Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***. The UMP Format is used for all data in the Clip Configuration Header and the Clip Sequence Data. See Section Sections ***3***, ***6***, and ***7***.

3.2 Delta Clockstamps

Two messages in the UMP format together form a mechanism for precise timing of events stored in a MIDI Clip File:

1. Delta Clockstamp Ticks Per Quarter Note (DCTPQ)
2. Delta Clockstamp (DCS)

For detailed definitions of these two messages, see Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]***.

3.2.1 Delta Clockstamp Ticks Per Quarter Note (DCTPQ)

The Delta Clockstamp Ticks Per Quarter Note message declares the unit of measure used by Delta Clockstamp messages in a MIDI Clip File.

A MIDI Clip File shall contain one DCTPQ message as the first message following the File Header and any Profile ID. The DCTPQ shall have a preceding Delta Clockstamp (see Section ***3.2.2***) with Number of Ticks Since Last Event set to 0x00000.

3.2.2 Delta Clockstamp (DCS): Ticks Since Last Event

The Delta Clockstamp message declares the time of all following messages which occur before the next Delta Clockstamp message.

The timing of every message (other than Delta Clockstamps) in a MIDI Clip File shall be set by the most recent preceding Delta Clockstamp. Simultaneous events may share a single Delta Clockstamp. The order of simultaneous events can be critical. Therefore, events shall always be stored and transmitted in presentation order.

If no MIDI message has occurred during the previous 1,048,575 ticks, then the application which creates the MIDI Clip File shall insert a Delta Clockstamp followed by a Null message to restart the delta time count. Then the next Delta Clockstamp in the file declares the ticks since the previous Null message.

Table 4 Maximum Times of Selected Ticks Per Quarter Note Values at Selected Tempos

		Number of Ticks Per Quarter Note (MIDI Clock = 24 TPQ)					
		1	**24**	**96**	**480**	**960**	**65,535**
	Max Number of Quarter Notes	1048575	43691	10923	2185	1092	16
Beats Per Minute	**Seconds Per Beat**	**Maximum Time Addressable at BPM Tempo**					
20	**3s 0ms**	873h 49m	36h 25m	9h 6m	1h 49m 14s	54m 37s	48s 1ms
60	**1s 0ms**	291h 16m	12h 8m	3h 2m	0h 36m 25s	18m 12s	16s 0ms
90	**0s 667ms**	194h 17m	8h 6m	2h 1m	0h 24m 17s	12m 9s	10s 672ms
120	**0s 500ms**	145h 38m	6h 4m	1h 31m	0h 18m 12s	9m 6s	8s 0ms
180	**0s 333ms**	97h 0m	4h 2m	1h 1m	0h 12m 7s	6m 4s	5s 328ms

3.3 Messages Replace the Former "Meta Events" in SMF v1

The Standard MIDI File version 1 specifications for Type 0 and Type 1 defined certain non-MIDI data as "Meta Events". MIDI 2.0 has a lot more space to define new MIDI messages, so these events have been replaced by new MIDI messages in the UMP Format. The Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol specification ***[MA06]*** defines many of those messages as Message Type D, Flex Data messages.

An application which reads these events in a MIDI Clip File may send these events to a Receiver, something that was not possible with the previous "Meta Events".

4 File Format

This section defines the basic file format and key data which are the foundation of the file.

A MIDI Clip File contains 3 sections:

1. File Header – The first 8 bytes in the file.
2. Clip Configuration Header – Data used to set up a Receiver to play the sequence.
3. Clip Sequence Data – A set of MIDI Messages in Universal MIDI Packet Format, presented in the main timeline of the sequence to be performed.

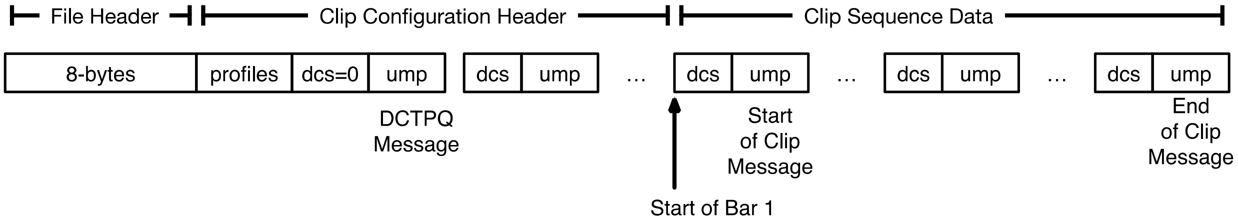


Figure 2 MIDI Clip File Structure

The File Header is defined in Section ***5*** of this document.

The Clip Configuration Header is defined in Section ***6*** of this document.

The MIDI Clip Sequence Data is defined in Section ***7*** of this document.

4.1 File Extensions

MIDI Clip Files shall use the file extension ".midi2".

5 File Header

A MIDI Clip File shall have an 8-byte File Header.

The File Header shall be the ASCII values to spell out "SMF2CLIP".

Table 5 MIDI Clip File Header Values

	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Value	0x53	0x4D	0x46	0x32	0x43	0x4C	0x49	0x50
Text	S	M	F	2	C	L	I	P

All data following the File Header are UMP messages. The ninth byte in the file, immediately following the File Header, shall be the first byte in the first UMP message in the Clip Configuration Header. See Section ***6***.

6 Clip Configuration Header

A MIDI Clip File shall have a Clip Configuration Header immediately following the 8-byte File Header.

All data in the Clip Configuration Header shall be UMP messages.

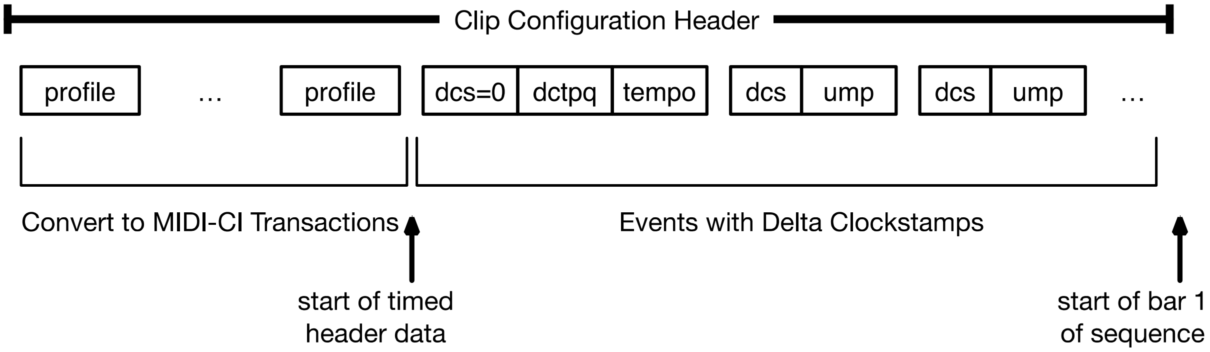

Clip Configuration Header
profile
…
profile
dcs=0
dctpq
tempo
dcs
ump
dcs
ump
…
Convert to MIDI-CI Transactions
Events with Delta Clockstamps
start of timed header data
start of bar 1 of sequence

Figure 3 Clip Configuration Header

If the MIDI Clip File includes Set Profile On messages (to indicate target Profiles for the sequence data, see Section 6.2), they shall be the first UMP messages in the Clip Configuration Header. Set Profile On messages shall not have a prepended Delta Clockstamp (DCS).

A Clip Configuration Header shall have a Delta Clockstamp Ticks Per Quarter Note message (DCTPQ, see Section ***3.2.1***) immediately following any optional Set Profile On messages or immediately following the File Header if there are no Set Profile On messages. The DCTPQ shall have a preceding Delta Clockstamp (DCS) with Number of Ticks Since Last Event set to 0x00000.

The Clip Configuration Header may contain other UMP messages as necessary to configure a Receiver such that the Receiver might properly execute UMP messages contained in the MIDI Clip Sequence Data. All these other UMP messages in the Clip Configuration Header shall have a prepended Delta Clockstamp (DCS) (see Section 3.2.2).

The Clip Configuration Header shall include all data after the File Header through to but not including the Delta Clockstamp (DCS) for the Start of Clip Message (See Section ***7.1***).

6.1 Configuration Timing: Delta Clockstamps, Tempo, and Time Signature

A Receiver might require some time to process certain configuration related messages. The time required to process Set Profile On messages is determined by the sequencer (see Section ***6.2***). Subsequent UMP messages in the Clip Configuration Header may be assigned a specific temporal location, using Delta Clockstamps and by including optional Set Tempo (See Section ***6.1.1***) and Set Time Signature (See Section ***6.1.2***) messages in the MIDI Clip Header.

The time the sequencer uses to process the Target Profile Identifiers (see Section ***6.2***) plus the cumulative values of all Delta Clockstamp up to and including the Delta Clockstamp for the Start of Clip Message determines the total length of time allocated to the Clip Configuration Header.

If the Clip Configuration Header includes a Set Time Signature message, the Set Time Signature message shall be the first event following the Set Tempo message.

The Clip Configuration Header shall not include more than one Set Tempo message and shall not include more than one Set Time Signature message.

6.1.1 Set Tempo Message in Clip Configuration Header

The Clip Configuration Header may contain a Set Tempo message as the first event following the DCTPQ message. This Set Tempo message in the Clip Configuration Header shall use the Delta Clockstamp which precedes the DCTPQ message or shall have a preceding Delta Clockstamp with Number of Ticks Since Last Event set to 0x00000

A sequencer may determine how many bits of resolution it will support to achieve the accuracy that the sequencer deems appropriate. A sequencer is not required to support all declared tempos on import; very slow or very fast tempos might be changed to the minimum or maximum of the sequencer at the time of importing.

The Clip Configuration Header shall not contain more than one Set Tempo message.

The Clip Sequence Data (see Section *7*) may contain multiple Set Tempo messages. When in Clip Sequence Data, Set Tempo messages should only occur at the same timing where MIDI Clock messages would be located.

6.1.2 Set Time Signature Message in Clip Configuration Header

A Clip Configuration Header may contain a Set Time Signature immediately following the Set Tempo message. This Set Time Signature shall use the Delta Clockstamp which precedes the DCTPQ message or shall have a preceding Delta Clockstamp with Number of Ticks Since Last Event set to 0x00000.

The Clip Configuration Header shall not contain more than one Set Time Signature message.

The Clip Sequence Data (see Section *7*) may contain multiple Set Time Signature messages. When in a Clip Sequence Data, Set Time Signature messages should only occur at the same timing as the beginning of a bar.

6.2 Receiver Configuration by MIDI-CI Profile Configuration

Some sequence data in a MIDI Clip File may be designated to be sent to a MIDI Device which supports a particular MIDI-CI Profile. MIDI-CI Profile Configuration may be used to configure a Receiver to optimally respond to sequence data.

If any sequence data is intended for a particular MIDI-CI Profile, a Set Profile On message should be included in the beginning of the Clip Configuration Header, with the appropriate Profile ID, to tag data as targeted to that Profile. Set the Destination Device ID field for the appropriate Channel, Group, or Function Block. Set the Source and Destination MUIDs to Broadcast.

> *Note: Authors of sequence data should be aware of the complexities of Profiles. Some Profiles are simple and consistent while others might have complex set of options for varied or optimum performance. Complex Profiles might require more user interaction with the sequencer or might be less likely to deliver a predictable outcome.*

6.2.1.1 Sequencer's Use of Profile Configuration Data

MIDI-CI requires bidirectional negotiations, but a MIDI Clip File inherently has no bidirectional mechanisms.

Therefore, the sequencer should read the Set Profile On message out of the Clip Configuration Header and perform a set of bi-directional MIDI-CI transactions to configure the Receiver. See ***[MA03]*** and ***[MA04]***. This entails at least the following steps:

1. Perform MIDI-CI Discovery (to get the MUID of the Receiver)
2. Perform MIDI-CI Profile Inquiry (to confirm the Receiver supports the Profile)
3. Send a Set Profile On message to the MUID of the Receiver

> *Note: This is the minimum set of steps. Some Profiles make use of more steps, including the use of the Profile Details Inquiry before the Set Profile On.*

The sequencer should perform this process those immediately upon loading the file, before starting to send the rest of the Clip Configuration Header and before the Clip Sequence Data is sent.

6.3 Receiver Configuration by Other MIDI Messages

The Clip Configuration Header may contain other MIDI messages in UMP Format to prepare the Receiver to properly perform the data in the Clip Sequence Data. Examples include but are not limited to Bank Select and Program Change, Control Changes to set Volume and Pan, Profile specific messages, and System Exclusive.

6.4 Property Exchange Resources – Not in the MIDI Clip File

Property Exchange (PE) Resources have complex and varied data sets for configuring a device. Property Data for PE Resources shall not be included in a MIDI Clip File. Applications that require use of Property Exchange in a Standard MIDI File shall use the MIDI Container File format instead (see Section ***2***).

7 Clip Sequence Data

The MIDI Clip File shall have a Clip Sequence Data section which follows the Clip Configuration Header.

All data in the Clip Sequence Data shall be UMP messages with associated Delta Clockstamps.

A Clip Sequence Data shall include one Start of Clip message (see Section *7.1*) with a preceding Delta Clockstamp as the first UMP message.

A Clip Sequence Data shall include one End of Clip message as the last UMP message (See Section *7.3*) with a preceding Delta Clockstamp.

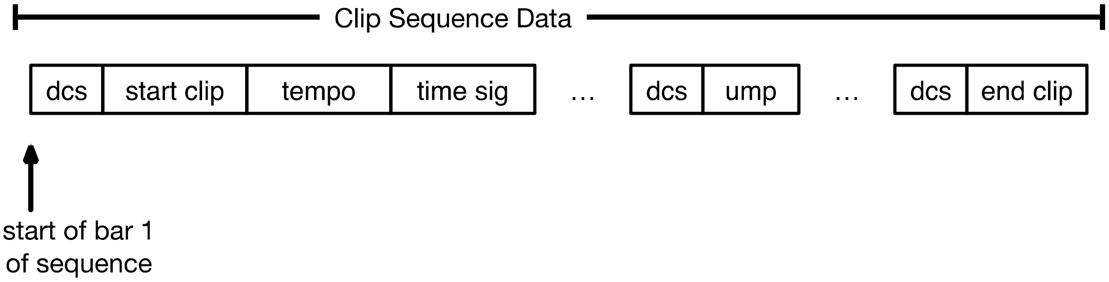


Figure 4 Clip Sequence Data

7.1 Start of Clip Message

The first event in the Clip Sequence Data shall be a Start of Clip message with a preceding Delta Clockstamp (DCS). If the Clip Sequence Data is used for musical content, then the timing of the Start of Clip message should be the start of the first bar of music.

7.1.1 Set Tempo Message

The Clip Sequence Data should contain a Set Tempo message at the start of the sequence, immediately following the Start of Clip message. This first Set Tempo message in the Clip Sequence Data should use the Delta Clockstamp which precedes the Start of Clip message or have its own Delta Clockstamp which has the same value as the value in the Delta Clockstamp which precedes the Start of Clip Message.

If the Clip Sequence Data is used for musical content, then the timing of the first Set Tempo message should be the start of the first bar of music. Any subsequent Set Tempo messages in the Clip Sequence Data should only occur where MIDI clocks would be located.

A sequencer may determine how many bits of resolution it will support to achieve the accuracy that the sequencer deems appropriate. A sequencer is not required to support all declared tempos on import; very slow or very fast tempos might be changed to the minimum or maximum of the sequencer at the time of importing.

7.1.2 Set Time Signature Message

A MIDI Clip File should contain a Set Time Signature message for the first bar of the sequence, immediately following the Start of Clip and Set Tempo messages. This first Time Signature message in the Clip Sequence Data should use the Delta Clockstamp which precedes the Start of Clip message.

Any subsequent Set Time Signature messages in the Clip Sequence Data should only occur at the time when one bar ends (according to the previously set time signature) and the next bar begins with the new time signature.

7.1.3 Pickup Bars

Some sequences have a "pickup bar", with a length which is different from the following bar. Such a bar should have a Set Time Signature message which accurately reflects the length of the pickup before the subsequent bar. The subsequent bar should start with a new Set Time Signature message.

7.2 MIDI Data

The Clip Sequence Data may contain any MIDI messages in UMP Format, except Property Exchange messages.

However, MIDI messages which are part of a bi-directional negotiation, such as MIDI-CI messages, might not be handled by the sequencer or performed by a Receiver in a predictable or meaningful manner.

7.3 End of Clip Message

The last event in the Clip Sequence Data shall be an End of Clip message with a preceding Delta Clockstamp. A MIDI Clip File shall not have any data following the End of Clip message.

Appendix A: SMF1 to SMF2 Concordance

This MIDI Clip File specification is part of a wider scope, an evolution from the Type 0 and Type 1 files defined in the prior Standard MIDI File version 1 specification to a design which supports MIDI 1.0 and MIDI 2.0.

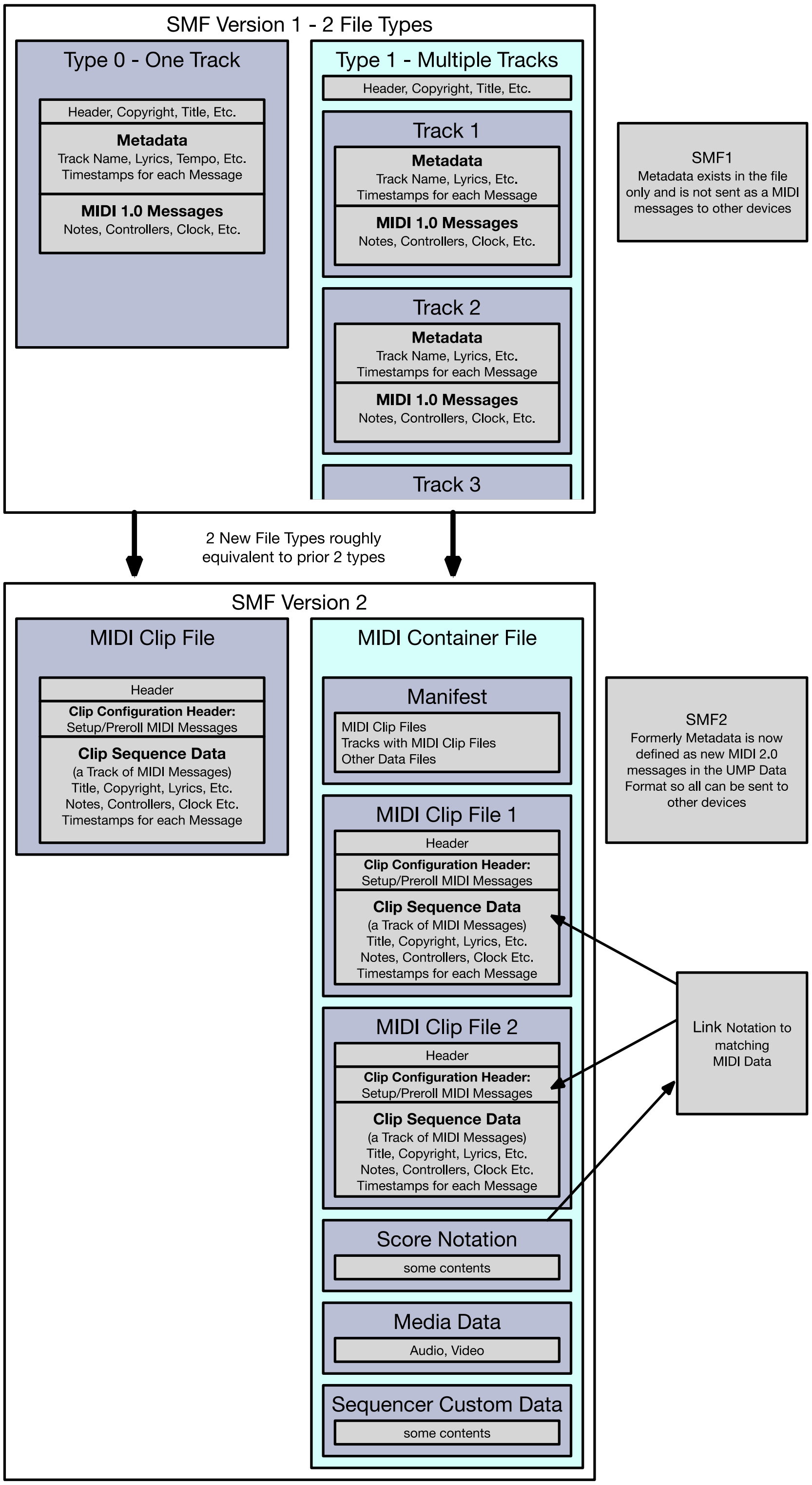

SMF Version 1 - 2 File Types
Type 0 - One Track
Header, Copyright, Title, Etc.
Metadata
Track Name, Lyrics, Tempo, Etc.
Timestamps for each Message
MIDI 1.0 Messages
Notes, Controllers, Clock, Etc.
Type 1 - Multiple Tracks
Header, Copyright, Title, Etc.
Track 1
Metadata
Track Name, Lyrics, Etc.
Timestamps for each Message
MIDI 1.0 Messages
Notes, Controllers, Clock, Etc.
Track 2
Metadata
Track Name, Lyrics, Etc.
Timestamps for each Message
MIDI 1.0 Messages
Notes, Controllers, Clock, Etc.
Track 3
SMF1
Metadata exists in the file only and is not sent as a MIDI messages to other devices
2 New File Types roughly equivalent to prior 2 types
SMF Version 2
MIDI Clip File
Header
Clip Configuration Header:
Setup/Preroll MIDI Messages
Clip Sequence Data
(a Track of MIDI Messages)
Title, Copyright, Lyrics, Etc.
Notes, Controllers, Clock Etc.
Timestamps for each Message
MIDI Container File
Manifest
MIDI Clip Files
Tracks with MIDI Clip Files
Other Data Files
MIDI Clip File 1
Header
Clip Configuration Header:
Setup/Preroll MIDI Messages
Clip Sequence Data
(a Track of MIDI Messages)
Title, Copyright, Lyrics, Etc.
Notes, Controllers, Clock Etc.
Timestamps for each Message
MIDI Clip File 2
Header
Clip Configuration Header:
Setup/Preroll MIDI Messages
Clip Sequence Data
(a Track of MIDI Messages)
Title, Copyright, Lyrics, Etc.
Notes, Controllers, Clock Etc.
Timestamps for each Message
Score Notation
some contents
Media Data
Audio, Video
Sequencer Custom Data
some contents
SMF2
Formerly Metadata is now defined as new MIDI 2.0 messages in the UMP Data Format so all can be sent to other devices
Link Notation to matching MIDI Data

Figure 5 Evolution from MIDI 1.0 Standard MIDI Files

Appendix B: MIDI Messages Useful for MIDI Clip Files.

Following are some MIDI messages from the UMP & MIDI 2.0 Protocol Specification ***[MA06]*** which are specifically designed for use with MIDI Clip Files. Many of these fill roles which were previously filled by Meta Events in SMF Version 1:

- Set Tempo
- Set Time Signature
- Set Metronome
- Extended MIDI Time Code
- SMPTE Offset
- Set Key Signature
- Set Chord
- Project Name
- Composition (Song) Name
- MIDI Clip Name
- Copyright Notice
- Composer Name
- Lyricist Name
- Arranger Name
- Publisher Name
- Primary Performer Name
- Accompanying Performer Name
- Recording/Concert Location
- Recording/Concert Date
- Lyrics

These messages are defined in the Universal MIDI Packet (UMP) Format and MIDI 2.0 Protocol ***[MA06]*** specification.

http://www.amei.or.jp

https://www.midi.org