SymphonyCSV: The Composer's Manual — Experimental Edition 0.3.2

Nov 2, 2026 · Red5Sorcery Books & Media

What changed in 0.3.2

Edition 0.3.2 extends SymphonyCSV from 4/4 to six meters (2/4, 3/4, 4/4, 5/4, 6/8 and 7/8) without adding, removing, renaming or reordering any of the 85 columns. The session-based workflow and pre-export regeneration from 0.3 are unchanged.

Area	Edition 0.3.1	Edition 0.3.2
Meters	4/4 only	2/4, 3/4, 4/4, 5/4, 6/8 and 7/8; one meter per piece
count_aloud	16 positions, 4/4 only	One defined vocabulary per meter, including One la li for 6/8
Measure length	Always 4 quarter notes	4 × numerator ÷ denominator quarter notes
onset_beat_in_measure	Beat number in 4/4	1 + quarter notes from the barline, in every meter
Tempo	Quarter notes per minute	Unchanged; its relation to compound and irregular pulses is documented
Support status	One verified path	Three tiers kept apart: defined, round-trip tested, verified on real music

The evidence so far: one synthetic test file per meter passed a MIDI round trip (section 8.5), and the generalized equations reproduce every Bach value except the running_time display already changed in 0.2. No meter other than 4/4 has yet been checked in MuseScore Studio 4 or exported to MusicXML. Earlier changes are in the revision history at the end.

1. Purpose, audience, and promises

SymphonyCSV lets a composer write music as rows in a spreadsheet and get back verified MIDI and MusicXML files. It is for people who think in measures, counts and notes, whether or not they play every part at a keyboard.

The composer is the author and decision-maker. An LLM may suggest ideas, explain theory, prepare rows, detect problems, regenerate the file and build exports. No musical change becomes authoritative until the composer approves it.

The file is ordinary UTF-8 comma-separated text with 85 columns in a fixed order. It opens in any common spreadsheet application.

1.1 What you can do without an LLM

- Read, annotate and edit the ten composer inputs in the first 12 columns.
- Choose notes, rhythms, velocities, instruments and every other musical idea.
- Open and review an already generated MIDI or MusicXML file.

1.2 What currently needs an LLM

In this experimental workflow, an LLM with code execution regenerates the 85 columns and builds the export files. A text-only chat can explain the process but cannot reliably deliver a working binary .mid file.

A deterministic converter could do this without an LLM, and working scripts exist for the Bach example. No distributed, general-purpose SymphonyCSV application exists yet.

> **Export claim, precisely stated.** Targets: Standard MIDI Files (MIDI 1.0, Format 1) and MusicXML 4.0-compatible files. Support covers only the musical features actually modelled and tested. Compatibility does not mean every MIDI or MusicXML feature is supported.

1.3 Planned publication

SymphonyCSV 1.0 is intended to become an openly documented format, with template, specification, example data and manual on a subpage of the Data Sculptor website. Neither the V1 release nor its licence has been announced. An open specification and open-source software are different claims; any code release needs its own explicit licence.

2. Two layers, one visible file

A composer writes ten inputs in the first 12 columns, plus a title and credit; regeneration fills everything else. All 85 columns always stay in the file, in the same order, so every value can be seen and checked.

2.1 The first 12 columns

One row per sounding note. A chord is several rows with the same measure and count.

#	Column	What you write	Example
01	staff	Which staff the note belongs to	Upper staff
02	measure	Bar number, starting at 1	3
03	count_aloud	Where the note starts, as you would count it	Two and
04	running_time	Leave for regeneration; shows where you are in the piece	0m 09s
05	pitch_label	Sounding pitch and octave (C4 = middle C)	F#5
06	duration_beats	Length in quarter-note beats	0.25
07	duration_ms	Leave for regeneration	250
08	velocity	Attack strength, 1 to 127 (64 if unsure)	90
09	tempo_bpm	Quarter notes per minute	60
10	time_signature	Meter: 2/4, 3/4, 4/4, 5/4, 6/8 or 7/8 (section 4.1)	4/4
11	key_signature	Key	C major
12	instrument	Part or instrument name	Piano

Ten of these are the composer inputs. running_time (04) and duration_ms (07) sit among them because they help you read the music, but they are calculated, never typed. The composer inputs, plus composition_title and composer in columns 73 and 74, are the source of truth for everything else.

2.2 The other 73 columns

Each of the remaining columns is filled in one of five ways. The exceptions are composition_title and composer (columns 73 and 74), which the composer writes once per piece. Appendix A gives the method for every column.

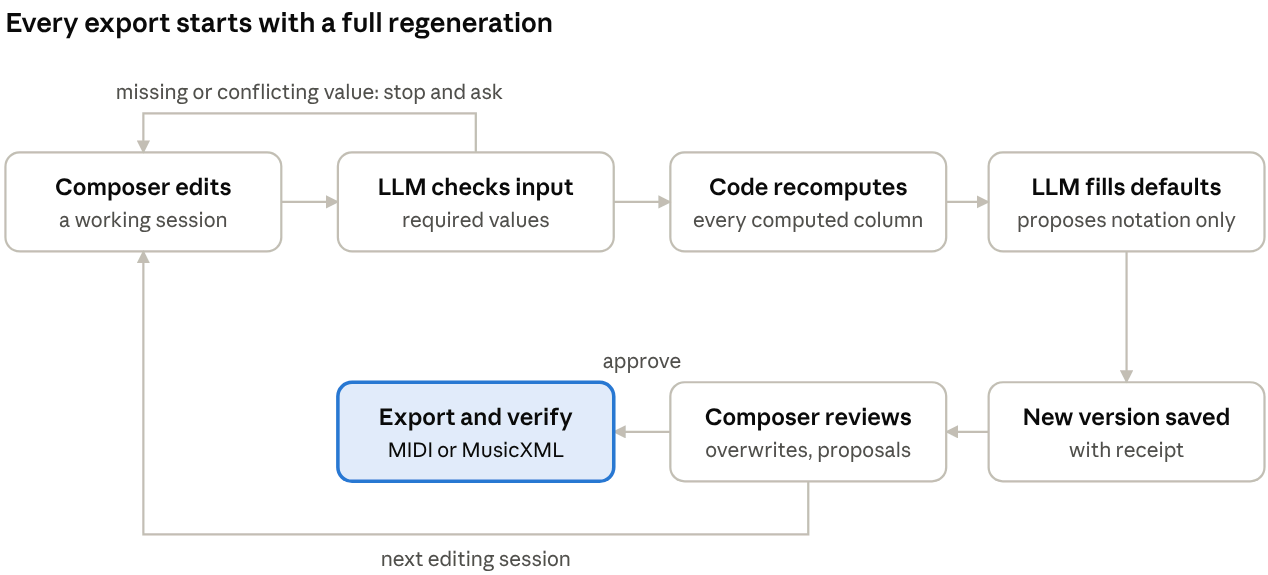
Filled by	What it means	Examples
Computed	Calculated by code from the composer inputs; never typed	midi_pitch, absolute_tick, end_ms
Mapped default	Derived from staff or part when blank; the composer may override	clef, midi_track, midi_channel
Policy	One setting for the whole piece, with a documented default	ticks_per_quarter, program_change_emit
Composer, optional	Blank unless the composer sets it; the LLM may only propose	voice, written_pitch, articulation, cc64_sustain
Source	Where a value came from, and its licence	source_note_data, source_license

2.3 Why everything stays visible

Nothing about the music is hidden in software. A reader can see the composer's intent, every calculated value, every policy and every provenance claim in one file. That visibility is what makes the regeneration and validation rules below checkable.

3. The pre-export regeneration rule

Compose freely, then regenerate once before each export. You may edit a single note or a whole section without a regeneration step each time. When a working session is ready for MIDI or MusicXML export, an LLM validates the whole CSV and regenerates every dependent field with deterministic code. Never export from stale calculations: a working draft may hold stale display or tick values, but an export-ready version may not.



regeneration cycle · 7 steps, 2 loops back to the composer

A problem in the input stops the cycle and returns to the composer; nothing is exported until the composer has reviewed the new version.

3.1 What regeneration does

1. **Check the composer inputs.** Every note needs a staff, measure, count, pitch, duration and velocity. A blank or invalid required value stops regeneration; the LLM asks rather than inventing a note.
2. **Check whole-piece values agree.** Tempo, meter, key, title and policies are repeated on every row. If any row disagrees, the LLM stops and asks; tempo and meter changes are not yet supported.
3. **Recompute every computed column** from the composer inputs, using deterministic code wherever available.
4. **Fill mapped defaults and policies** that are blank, using the documented defaults. Values the composer set are kept.
5. **Leave optional composer columns alone.** The LLM never fills voice, written_pitch, articulation, ties, beams, controllers, lyrics or markers on its own.
6. **Write a new version** of the complete 85-column file, with a new version number. The previous version is kept.
7. **Issue a regeneration receipt** (section 8.2).

3.2 Who wins a conflict

The composer inputs are the truth. If a computed column disagrees with them, regeneration overwrites it.

Every overwritten value is listed in the receipt by event_id, column, old value and new value. Nothing is overwritten silently.

If an overwrite looks deliberate, the LLM says so. Example: duration_ms was changed by hand but duration_beats was not. The composer probably meant to change the note length, so the LLM suggests making that change in duration_beats instead.

3.3 Computation and judgment are different jobs

Arithmetic is done by code; judgment is done by the LLM and recorded. Ticks, milliseconds, MIDI pitch numbers and running time have one right answer. An LLM working through hundreds of rows by hand will eventually slip a digit, so these are computed with code execution. If no code tool is available, the receipt must say the values were computed without one.

Judgment calls include voice assignment, enharmonic spelling, beaming and how to notate overlapping notes. The LLM may propose these, with reasons. Proposals live in the receipt, not in the CSV, until the composer adopts them. Adopted values are written into the optional composer columns and become the composer's decisions.

3.4 Do not hand-edit computed columns

You may type anywhere, but changes to computed columns will be overwritten at the next regeneration. To change the music, change the composer inputs. To change a whole-piece setting, change the policy column on every row, or ask the LLM to do it during regeneration.

3.5 Working drafts and refreshing

Between exports, the file is a working draft. After any edit that adds, removes or moves notes, its computed columns, including running_time, may be out of date.

If you want current display values while composing, ask for a refresh (prompt B.2). A refresh recalculates the computed columns and lists what changed. It is not an export gate: the full pre-export regeneration in Appendix D still runs before every export.

4. Meters, counts, running time, and precise time

Every timing value comes from measure, count_aloud and duration_beats, measured in quarter notes and converted to MIDI ticks. Ticks are the timing authority; milliseconds and running_time are display values. Edition 0.3.2 supports six meters, one meter per piece.

4.1 Simple, compound and irregular meters

A time signature's top number counts units and its bottom number names the unit: 4 means quarter notes, 8 means eighth notes. What a listener feels as the beat depends on how those units group.

- **Simple meter:** each beat divides into two. 2/4, 3/4, 4/4 and 5/4 have a quarter-note beat, split into two eighths or four sixteenths.
- **Compound meter:** each beat divides into three. 6/8 has two beats, each a dotted quarter made of three eighths. It feels like two, not six.
- **Irregular meter:** the beats are unequal. 7/8 groups its seven eighths as 2+2+3, 3+2+2 or 2+3+2. SymphonyCSV counts every eighth, so timing never depends on the grouping.

Meter	Kind	Felt beats	Quarter notes per measure	Ticks per measure (384 PPQ)	Count positions
2/4	Simple	2 quarters	2	768	8
3/4	Simple	3 quarters	3	1,152	12
4/4	Simple	4 quarters	4	1,536	16
5/4	Simple	5 quarters, often 3+2	5	1,920	20
6/8	Compound	2 dotted quarters	3	1,152	12
7/8	Irregular	3 unequal groups, e.g. 2+2+3	3.5	1,344	14

3/4 and 6/8 last the same time, three quarter notes, but feel different: three beats of two eighths against two beats of three. The counting vocabulary is what keeps them apart.

4.2 Counting aloud

count_aloud stays text, and each meter has exactly one vocabulary. A count is valid only if it appears in its meter's table; anything else stops regeneration. Every vocabulary sits on a sixteenth-note grid: 0.25 quarter notes, or 96 ticks at 384 PPQ.

Simple meters (2/4, 3/4, 4/4, 5/4). Beats are One to Five, and sixteenths are e, and, a. A count has the same offset in every simple meter; each meter uses the beats up to its top number.

Count	Offset (quarter notes)	Offset (ticks)	Valid in
One	0	0	2/4, 3/4, 4/4, 5/4
One e	0.25	96	2/4, 3/4, 4/4, 5/4
One and	0.5	192	2/4, 3/4, 4/4, 5/4
One a	0.75	288	2/4, 3/4, 4/4, 5/4
Two	1	384	2/4, 3/4, 4/4, 5/4
Two e	1.25	480	2/4, 3/4, 4/4, 5/4
Two and	1.5	576	2/4, 3/4, 4/4, 5/4
Two a	1.75	672	2/4, 3/4, 4/4, 5/4
Three	2	768	3/4, 4/4, 5/4
Three e	2.25	864	3/4, 4/4, 5/4
Three and	2.5	960	3/4, 4/4, 5/4
Three a	2.75	1,056	3/4, 4/4, 5/4
Four	3	1,152	4/4, 5/4
Four e	3.25	1,248	4/4, 5/4
Four and	3.5	1,344	4/4, 5/4
Four a	3.75	1,440	4/4, 5/4
Five	4	1,536	5/4
Five e	4.25	1,632	5/4
Five and	4.5	1,728	5/4
Five a	4.75	1,824	5/4

Compound meter (6/8). Two beats, One and Two. Each beat holds three eighths, counted beat, la, li, so a run of eighths reads One la li, Two la li. A sixteenth between eighths adds ta.

Count	Offset (quarter notes)	Offset (ticks)
One	0	0
One ta	0.25	96
One la	0.5	192
One la ta	0.75	288
One li	1	384
One li ta	1.25	480
Two	1.5	576
Two ta	1.75	672
Two la	2	768
Two la ta	2.25	864
Two li	2.5	960
Two li ta	2.75	1,056

Irregular meter (7/8). Count every eighth, One to Seven; the sixteenth between eighths is and. Under a 2+2+3 grouping, the groups begin on One, Three and Five.

Count	Offset (quarter notes)	Offset (ticks)
One	0	0
One and	0.25	96
Two	0.5	192
Two and	0.75	288
Three	1	384
Three and	1.25	480

Count	Offset (quarter notes)	Offset (ticks)
Four	1.5	576
Four and	1.75	672
Five	2	768
Five and	2.25	864
Six	2.5	960
Six and	2.75	1,056
Seven	3	1,152
Seven and	3.25	1,248

Grouping is notation, not timing. To record it, add it after the meter in time_signature, as in 7/8 (2+2+3) or 5/4 (3+2). Regeneration reads only the meter for timing; an exporter may use the grouping for beaming.

No vocabulary includes triplets in simple meters, thirty-second notes, or the simple-meter syllables e, and, a in 6/8. Each meter uses only its own words, so a count always has exactly one meaning.

4.3 Timing equations

With Q = quarter notes per measure, p = the count's offset from its table, m = measure and PPQ = ticks_per_quarter:

- Q = 4 × time_signature_num ÷ time_signature_den
- onset_beat_in_measure = 1 + p, always in quarter notes, in every meter
- absolute quarters = (m − 1) × Q + p
- onset_tick_in_measure = p × PPQ
- absolute_tick = absolute quarters × PPQ
- duration_ticks = duration_beats × PPQ
- end_tick = absolute_tick + duration_ticks
- composition_end_tick = composition_total_measures × Q × PPQ
- onset_ms = round(absolute quarters × 60,000 ÷ tempo_bpm); duration_ms and end_ms the same way

Compute with exact fractions or integers and round only once, at the end. At 384 PPQ every count in every vocabulary, and every multiple of a sixteenth, lands on a whole tick. A duration that does not, such as a triplet value of 1/3, is rounded to the nearest tick under tick_rounding_policy and listed in the receipt.

For 4/4, Q is 4 and these equations give exactly the earlier values. Regenerating the Bach master with them changes no tick, millisecond or pitch value; only the 266 running_time displays from the 0.2 rounding rule change.

Never derive note starts from running_time, and never add rounded duration_ms values together. Those columns are human-readable checks, not the timing authority.

4.4 Tempo in compound and irregular meters

tempo_bpm stays quarter notes per minute and tempo_beat_unit stays quarter, matching MIDI, which always stores tempo per quarter note. Other tempo_beat_unit values are not supported in 0.3.2. In /8 meters the felt pulse is not the quarter note, so it runs at a different rate.

Meter	Felt pulse	Pulses per minute	At tempo_bpm 96
2/4, 3/4, 4/4, 5/4	Quarter	tempo_bpm	96 quarters
6/8	Dotted quarter	tempo_bpm ÷ 1.5	64 dotted quarters
7/8	Eighth	tempo_bpm × 2	192 eighths

To set a 6/8 piece at 72 dotted-quarter beats a minute, enter tempo_bpm 108 (72 × 1.5). Durations stay in quarter notes in every meter: in 6/8 an eighth is 0.5, a dotted quarter 1.5 and a full bar 3.

4.5 Worked examples

Each row was computed by the reference regenerator at 384 PPQ.

Meter and tempo	Note	Q	p	Absolute quarters	absolute_tick to end_tick	onset_ms	running_time
2/4, 60 BPM	Measure 7, Two e, 0.25	2	1.25	13.25	5,088 to 5,184	13,250	0m 13s
3/4, 90 BPM	Measure 5, Three and, 0.5	3	2.5	14.5	5,568 to 5,760	9,667	0m 09s

Meter and tempo	Note	Q	p	Absolute quarters	absolute_tick to end_tick	onset_ms	running_time
5/4, 88 BPM	Measure 10, Five a, 0.25	5	4.75	49.75	19,104 to 19,200	33,920	0m 33s
6/8, 96 BPM	Measure 3, Two la, 0.5	3	2	8	3,072 to 3,264	5,000	0m 05s
7/8 (2+2+3), 120 BPM	Measure 2, Five, 1.5	3.5	2	5.5	2,112 to 2,688	2,750	0m 02s

The 3/4 note ends at tick 5,760, exactly the end of measure 5, so it is valid; one tick longer would cross the barline. In 6/8, Two la sits 2 quarter notes into the bar: one dotted-quarter beat (1.5) plus one eighth (0.5). The 7/8 note fills the final group of three eighths and ends on the barline (2 × 1,344 = 2,688).

4.6 running_time: where you are in the piece

running_time shows the elapsed time at each note's start as Xm YYs, rounded down to the whole second already reached. A note starting at 32.5 seconds shows 0m 32s, the way a clock or a DAW timeline reads.

- running_time = whole seconds in floor(onset_ms / 1000), shown as Xm YYs

Rounding down is a deliberate rule, and every tool must use it. Edition 0.1 left the rule unstated, and the Bach v01 master rounded exact halves to the nearest even second. A spreadsheet's ROUND function would have disagreed on 6 of 549 rows. Regenerating the Bach master under the 0.2 rule changes running_time on 266 rows and changes no note data.

At 60 BPM, four sixteenth notes share each displayed second. That is expected: running_time is a place marker, not a timestamp. In a working draft it may be stale until the next refresh or pre-export regeneration (section 3.5). composition_duration uses the same rounding-down rule.

4.7 Chords, overlaps, and notes across bars

A chord is several rows with the same onset. Notes may overlap; on one staff this may need more than one written voice in MusicXML, though MIDI needs nothing special. Every note must end at or before its barline: p + duration_beats ⩽ Q. Ties across a barline, changes of tempo or meter within a piece, pickup measures, tuplets and repeats are not yet supported.

5. Composing and collaborating with an LLM

Write the ten composer inputs and a title and credit, revise as often as you like, and hand the complete file to an LLM with Appendix D when you are ready to export. The steps below are the whole workflow.

1. **Make a working copy.** Save the template as a new UTF-8 CSV. Keep the 85 headers and their order. Delete the Bach note rows; keep Bach as a separate reference file.
2. **Decide the piece.** Choose a title, your composer credit, a tempo, one meter from section 4.1, a key or tonal plan, and the instruments. Put the title and credit in columns 73 and 74.
3. **Write and revise.** For each note, fill staff, measure, count_aloud, pitch_label, duration_beats and velocity. Fill tempo_bpm, time_signature, key_signature and instrument on every row. Edit as much as you like; ask for a refresh (B.2) whenever you want current display values.
4. **Prepare and export.** When the session is done, give the LLM the complete CSV, this manual and the Appendix D prompt. It regenerates, validates, exports and verifies in one pass.
5. **Listen, inspect, revise.** Open the export, listen, and edit the composer inputs. The next export runs Appendix D again.

5.1 A four-note teaching example

This shows the shape of four note events in measure 1. It is an exercise, not a claim of an original work.

staff	measure	count_aloud	pitch_label	duration_beats	velocity
Upper staff	1	One	C4	1	64
Upper staff	1	Two	E4	1	64
Upper staff	1	Three	G4	1	64
Upper staff	1	Four	C5	1	64

After regeneration, each of these rows carries all 85 columns.

5.2 Protect your creative authority

Before an LLM changes music, say what is protected: melody, bass line, harmony, tempo, density, rhythm, instruments or particular measures. Ask for a change list with event IDs and old and new values. A note change you did not authorize is a suggestion, not an edit.

5.3 What to give the LLM

- This manual, in a form the model can read.
- The complete 85-column CSV, not a screenshot or the first few rows.
- The task: compose a bounded phrase, edit rows, regenerate, export, or compare files.
- Your constraints: measures, meter, tempo, instrument, note range, forbidden edits, intended mood or harmony.
- Permission to use code execution and file tools.

5.4 Standing instruction for any LLM

> You are assisting a human composer using SymphonyCSV Experimental 0.3.2. Read this manual and the complete 85-column CSV. The composer inputs, title, credit, policies and any optional values the composer filled are authoritative. Allow unrestricted editing sessions. Check every count against the vocabulary for the piece's meter, and take measure length from time_signature_num and time_signature_den. Before export, validate required inputs and regenerate computed fields with deterministic code wherever available; never estimate ticks or milliseconds by hand. Fill mapped defaults and policies only where they are blank. Preserve stable event IDs, intentional blanks and the exact 85-column order. Never fill optional notation or controller values on your own; propose instead. Stop and ask about unresolved conflicts. Produce a versioned, export-ready CSV with a regeneration receipt. Export only supported MIDI or MusicXML features, parse each file back independently, and report the note-event comparison.

5.5 Reliability rules

- A confident explanation is not proof a file was built. Ask for the file and the verification output.
- An LLM must not reorder rows, drop notes, or replace unusual musical choices with "better" ones.
- Keep each accepted version as its own file, with its receipt.

6. MIDI export

The target is a Standard MIDI File, Format 1, MIDI 1.0, holding only events the CSV specifies. Format 1 puts tempo and meter on a conductor track (track 0) and notes on their own tracks.

6.1 Exporter discipline

An exporter makes no musical or technical decision of its own. Every event in the file must trace to a CSV value or a documented policy. A convenience the composer wants, such as a piano program change, belongs in the CSV, not in the exporter.

- **Program change.** Emit one only when program_change_emit is Yes. If you want receiving software to load a piano, set program_change_emit to Yes in a new version of the file. The exporter must not add it on its own.
- **Key signature.** Emit a key-signature meta event from key_signature on the conductor track when key_signature is filled. source_key_signature records whether a source MIDI had one. A reproduction that must match its source byte for byte may omit it, and the receipt says so.
- **Time signature and tempo.** Always emitted on the conductor track at tick 0. The time-signature event takes its numbers from time_signature_num and time_signature_den. Its metronome-click field follows a fixed policy: a quarter note (24 MIDI clocks) for /4 meters, a dotted quarter (36) for 6/8 and an eighth (12) for 7/8. The click guides metronomes only and never changes note timing. Tempo is always written per quarter note.
- **Track ends.** Every track, including the conductor track, ends at composition_end_tick, so all tracks report the same length.
- **Controllers.** A blank controller cell means no event. Zero is an explicit value. If a controller column is filled but the exporter does not implement it, the export stops and says so; it does not skip it silently.
- **Whole-piece values.** If tempo, meter or any policy differs between rows, the export stops. Tempo and meter changes are not yet supported.

6.2 Field mapping

CSV fields	MIDI result
absolute_tick, end_tick	Note on and note release positions (computed from measure, count and duration)
midi_pitch	Note number 0 to 127 (computed from pitch_label)
velocity	Note-on velocity 1 to 127
midi_track, midi_track_name, midi_channel	Track placement, track name, channel 1 to 16
midi_program, program_change_emit	Program change only when emit is Yes
note_off_method, note_off_velocity	Release as Note Off, or Note On with velocity 0
same_tick_event_order	Releases before attacks at the same tick
tempo_us_per_quarter, time_signature_num/den, key_signature_fifths/mode	Conductor-track meta events

6.3 What MIDI does not contain

A MIDI file holds musical events, not sound. It does not carry the receiving application's instrument preset, effects, mix or page layout. The same file can sound different in different software.

7. MusicXML, MuseScore Studio 4, and DAWs

MusicXML carries written notation, so it exposes every notation decision the CSV leaves blank. Pitches, onsets, durations and velocities are exact; clefs, voices, beams, rests and spellings are exact only where the CSV states them.

7.1 What is exact and what is inferred

Information	Source of truth
Pitch, onset, sounding duration, velocity	The composer inputs, after regeneration
Clef and staff	Explicit when set; otherwise mapped from staff
Voice and beaming	Optional composer columns when set; otherwise inferred at export
Written spelling, ties, articulations	Only when set; inferences are labelled as inferences
Title and composer	Metadata columns
Page layout, complex directions	Not yet modelled

Inferences made at export time are listed in the export receipt, not written back into the CSV. The composer may adopt them, and only then do they become part of the file.

Meter in MusicXML. An export writes the meter as beats and beat-type, for example 6 and 8. Beaming should follow the meter's grouping: eighths in threes for 6/8, and the declared grouping for 7/8 or 5/4. Without a grouping suffix, beaming is inferred and listed in the receipt. No meter other than 4/4 has yet been exported to MusicXML from the 85-column master.

7.2 Known notation issues in the Bach master

- **Sharp spellings.** Pitch labels were taken from MIDI note numbers and always use sharps (A#3, D#4, G#3 and others). The source score may spell some of these as flats. written_pitch is blank, so a MusicXML export will show sharps unless the composer fills it.
- **Staff follows the MIDI track.** The Upper staff comes from the source's upper MIDI track. From measure 16 it dips below middle C, reaching D3 in measure 33. On a treble clef with no clef change, that means many ledger lines. The schema has no mid-staff clef-change rule yet.

7.3 MusicXML limits

The 85 columns do not cover all of MusicXML 4.0. Slurs, tuplets, grace notes, layout, multiple verses, tablature, percussion and microtonal notation need richer modelling. Well-formed MusicXML does not guarantee that every application keeps its layout or playback details.

7.4 Checking in MuseScore Studio 4

MuseScore Studio 4 is the reference notation application for this guide; SymphonyCSV does not depend on it.

1. Open the .musicxml or .mid file with File > Open.
2. Check tempo, meter, measure count, both staves, and the opening and closing notes.
3. Play it and listen for wrong pitches, missing attacks, retriggers or short notes.
4. If the music is right but looks different, check the voice, beam, clef and notation columns before touching pitches or timing.
5. Save engraving work in MuseScore's own format if you want to keep it.

Prefer MusicXML for notation. A MIDI import makes MuseScore guess clefs, voices, note values, spellings and rests. If a score looks wrong, check MuseScore's import settings before blaming the data.

7.5 Using exports in a DAW

Import the .mid file into any DAW that reads Standard MIDI Files, check the tracks and tempo, and assign instruments. Fix note data in the CSV; the next export regenerates it. Sound design and mixing stay in the DAW project, because MIDI and MusicXML do not store them. Some DAWs also import MusicXML; test that rather than assume it.

8. Validation and receipts

A file passes only when it can be parsed back and its note events match the composer inputs exactly; a file that opens and looks right is not enough. Every refresh, pre-export regeneration and export produces a receipt.

8.1 Checks

Check	Pass criterion
Schema	Exactly 85 headers in the documented order, UTF-8
Required input	Every note has staff, measure, count, pitch, duration and velocity
Whole-piece agreement	Tempo, meter, key, title and policies identical on every row
Supported meter	time_signature is 2/4, 3/4, 4/4, 5/4, 6/8 or 7/8, with an optional grouping suffix
Count vocabulary	Every count_aloud appears in its meter's table (section 4.2)
Measure fit	Every note ends at or before its barline: p + duration_beats ⩽ Q
Computed columns	Every computed value matches a fresh calculation from the composer inputs
Tick representation	Every onset and duration is a whole number of ticks, or its rounding is listed in the receipt
running_time rule	Every value equals the onset rounded down to whole seconds
Chronology	No negative durations; overlaps intentional
Event IDs	Unique; existing IDs kept across edits
MIDI structure	Parser reads the header, every track and the conductor meta events
MIDI meter	The time-signature event matches time_signature_num and time_signature_den, with the click policy in 6.1; every track ends at composition_end_tick
MIDI events	Exact match of (track, channel, onset tick, pitch, velocity, duration ticks) for every note
MusicXML structure	Well-formed; schema-valid where a validator is available

Check	Pass criterion
MusicXML events	Sounding notes rebuilt from the XML match every note in the CSV
Human review	The composer listens, inspects and approves

8.2 Regeneration receipt

- Source file and new version name
- Tool used for computation (code execution, or none)
- Notes checked, and any that stopped regeneration
- Every overwritten value: event_id, column, old value, new value
- Defaults and policies filled, and their values
- Notation proposals offered, not applied

8.3 Export receipt

- Source CSV version and export type
- Exporter and tool environment
- MIDI policies applied (program change, key signature, note release method)
- Notes expected and notes parsed back; missing and extra counts
- Inferences made (voices, beams, rests, spellings), labelled as inferences
- Columns present but not exported

If a check fails, there is no success receipt. The receipt names the event_id and the conflicting columns. If a file cannot be built, say so; never substitute a sample file. Keep the previous CSV and export until the new one passes.

8.4 The Bach checkpoint

The 85-column Bach master has been exported to MIDI and verified on all 549 notes, with zero missing and zero extra.

Receipt field	Result
Source	SymphonyCSV_Bach_Prelude_C_BWV846_Master_60_BPM_v01.csv

Receipt field	Result
Target	MIDI 1.0 Standard MIDI File, Format 1, 384 ticks per quarter
Tracks	Conductor (0), lower staff on channel 1 (1), upper staff on channel 2 (2)
Notes expected / parsed	549 / 549; 0 missing, 0 extra, 0 unterminated
Comparison	Parsed back with an independent MIDI library; full note-tuple match
Cross-checks	A second, independently produced export and the original Mutopia MIDI: all 549 notes match
Length	Every track ends at tick 53,760 (2m 20s)
Policies	No program change (program_change_emit = No); note release as Note On, velocity 0
Assumption	Key signature C major emitted; the source MIDI had none

This proves the 85-column master reproduces Bach exactly through MIDI. It does not yet test program changes, Note Off releases, controllers, tempos other than 60, or MusicXML export from the 85-column master.

8.5 Meter round-trip tests

One synthetic three-measure test file per meter was regenerated, exported to MIDI and parsed back, and all six passed. These tests check the timing rules and the MIDI path; they do not check notation or real music.

Each file holds every count position of its meter once, as sixteenths, in measure 1. Measure 2 is a three-note chord lasting the whole bar. Measure 3 has one note per felt beat, or per group for 7/8 (2+2+3).

Meter	Tempo (BPM)	Notes	Ticks per measure	MIDI click	Result
2/4	72	13	768	24	Pass
3/4	90	18	1,152	24	Pass
4/4	88	23	1,536	24	Pass

Meter	Tempo (BPM)	Notes	Ticks per measure	MIDI click	Result
5/4	100	28	1,920	24	Pass
6/8	96	17	1,152	36	Pass
7/8 (2+2+3)	120	20	1,344	12	Pass

Pass means four things held. Every note matched on (track, channel, onset tick, pitch, velocity, duration ticks) after parsing with an independent MIDI library. Each count landed on its expected 96-tick position. The time-signature event matched the meter and click policy. Every track ended at exactly three measures.

The rejections were also checked. Counts from the wrong vocabulary (Three or One e in 6/8, Four in 3/4) and a 7/8 note crossing its barline each stopped regeneration with a message naming the event.

Not yet tested: opening these files in MuseScore Studio 4, MusicXML export in any meter other than 4/4, and real music in these meters. The test files and reference scripts accompany this edition.

9. Scope, open decisions, and planned V1

What exists today is a verified Bach master and MIDI path; what remains is breadth, a general converter, and a few open decisions.

9.1 What exists

- The 85-column Bach master, 35 measures, 549 notes, with technical and provenance fields.
- A verified 85-column MIDI export of that master (section 8.4).
- A reference regenerator and MIDI exporter, with six synthetic meter test files that pass a MIDI round trip (section 8.5).
- A 12-column compact predecessor, used for an earlier verified CSV-to-MIDI reconstruction.
- An experimental CSV-to-MusicXML demonstration whose sounding notes were checked.
- This manual.

9.2 Not yet a released capability

- A packaged end-user application or a general converter for all 85 columns.
- Tempo and meter changes within a piece, pickup measures, tuplets, notes across barlines, repeats, mid-staff clef changes.
- Full MusicXML 4.0 coverage, or every MIDI 1.0 message.
- An announced licence, public V1 specification or official website.

9.3 Meter support status

The schema and counting rules now cover six meters, but only 4/4 has been verified on real music. This edition keeps three levels apart: what the schema can describe, what has passed a deterministic round trip, and what has been verified on real music and in notation software.

Meter	Schema and counts defined	Regeneration and MIDI round trip	Real music through MIDI	MuseScore Studio 4	MusicXML from 85 columns
4/4	Yes	Passed (synthetic, 8.5)	Verified: Bach, 549/549 notes	Reported informally by the authors; no receipt	Not yet tested
2/4, 3/4, 5/4	Yes	Passed (synthetic, 8.5)	Not yet tested	Not yet tested	Not yet tested
6/8	Yes	Passed (synthetic, 8.5)	Not yet tested	Not yet tested	Not yet tested
7/8	Yes	Passed (synthetic, 8.5)	Not yet tested	Not yet tested	Not yet tested

Changes of meter within one composition are a separate future capability. The schema could hold a different time_signature on each row, but regeneration and export stop when rows disagree. Supporting changes needs its own rules for measure lengths, counting and MIDI meta events, and its own tests.

Meters beyond these six, such as 3/8, 9/8, 12/8 or 2/2, have no defined vocabulary yet. They would follow the same patterns but are not part of 0.3.2.

9.4 Open decisions for the composer

- [ ] Confirm the running_time rule: rounding down to whole seconds (current rule), or tenths of a second (0m 32.5s) for finer placement.

- [ ] Confirm the key-signature rule in 6.1: emit when key_signature is filled.
- [ ] Regenerate the Bach master as v02 under the rounding-down running_time rule (266 display values change; no note data).
- [ ] Confirm the 7/8 convention: count every eighth (this edition) rather than counting groups.
- [ ] Confirm the grouping suffix in time_signature, such as 7/8 (2+2+3), as the way to record grouping.
- [ ] Confirm the MIDI click policy: 24 for /4 meters, 36 for 6/8, 12 for 7/8.
- [ ] Decide whether the LLM may adopt notation proposals in bulk on request, or only row by row.
- [ ] Decide how a future version records which optional values the composer approved, without changing the 85-column order.

9.5 Change control

The 85-column order is frozen for the experimental baseline. Any schema change needs a version number, a migration note, example files, and tests showing note events are unchanged.

9.6 Evidence gates for V1

- Publish the exact schema: every column's type, range, blank meaning, precedence and filler.
- Test original piano pieces at more than one tempo, with chords, rests, overlaps and two staves.
- Open each meter test file in MuseScore Studio 4 and record what it shows.
- Write and export at least one real piece in a meter other than 4/4, including 6/8.
- Test the untested MIDI paths: program change Yes, Note Off releases, sustain pedal and other controllers.
- Show MusicXML export from the 85-column master in several meters, with structure validation and note-event comparison.
- Test import and playback in MuseScore Studio 4 and at least one DAW, recording what differs.
- Specify and test changes of meter within a piece as a separate capability.

9.7 Ownership, rights, and privacy

The composer decides what is written, shared, revised and published. Before uploading unpublished work to an online LLM, check the service's file handling and privacy settings. An imported MIDI or MusicXML file does not grant redistribution rights; keep its attribution and licence in the source columns.

Appendix A. The 85-column dictionary

Every column, in file order, with who fills it and what a blank means. "Composer, optional" columns stay blank unless the composer sets them; the LLM may propose values but never fills them on its own. Controller columns 58 to 71 are not yet implemented by a tested exporter, so a filled cell stops the export with a message.

#	Column	Filled by	Type	Meaning and blank rule
01	staff	Composer	Text	Staff for the note, e.g. Upper staff. Required
02	measure	Composer	Integer	Bar number from 1; measure 1 starts at time zero. Required
03	count_aloud	Composer	Text	Spoken onset in the bar, from its meter's vocabulary (section 4.2), e.g. Two and, One la, Five. Required (changed in 0.3.2)
04	running_time	Computed	Text	Onset rounded down to whole seconds, as Xm YYs. Display only (changed in 0.2)
05	pitch_label	Composer	Text	Sounding pitch and octave; C4 = MIDI 60. Required
06	duration_beats	Composer	Decimal	Length in quarter beats; 0.25 = sixteenth. Required
07	duration_ms	Computed	Integer	duration_beats × 60,000 / tempo, rounded. Display only
08	velocity	Composer	Integer	Attack 1 to 127; 0 is reserved for release. 64 if unsure; Bach uses 90
09	tempo_bpm	Composer	Decimal	Quarter notes per minute; same on every row

#	Column	Filled by	Type	Meaning and blank rule
10	time_signature	Composer	Text	2/4, 3/4, 4/4, 5/4, 6/8 or 7/8, with an optional grouping suffix such as 7/8 (2+2+3); same on every row (changed in 0.3.2)
11	key_signature	Composer	Text	e.g. C major. Affects spelling and the key meta event, not MIDI pitch
12	instrument	Composer	Text	Part or instrument name; Piano if blank
13	event_type	Mapped default	Enum	note; other event types not yet supported
14	event_id	Computed	Text	Unique ID, N00001 onward; existing IDs kept across edits
15	onset_beat_in_measure	Computed	Decimal	1 + quarter notes from the barline, in every meter: One = 1; in 6/8, Two = 2.5 (changed in 0.3.2)
16	onset_ms	Computed	Integer	Onset from the start of the piece, rounded. Display only
17	end_ms	Computed	Integer	Release time in ms, rounded. Display only
18	midi_pitch	Computed	Integer	Note number 0 to 127 from pitch_label
19	velocity_description	Computed	Text	Label such as Medium; bands to be documented before V1
20	written_pitch	Composer, optional	Text	Spelling to engrave, e.g. Bb3 for A#3. Blank: none asserted
21	pitch_spelling_source	Source	Text	Where spelling came from: score, composer or MIDI inference
22	clef	Mapped default	Text	Treble for upper, Bass for lower, unless set
23	clef_sign	Mapped default	Text	G or F, from clef
24	clef_line	Mapped default	Integer	2 for G, 4 for F
25	voice	Composer, optional	Integer	Written voice on its staff. Blank: inferred at export, listed in receipt

#	Column	Filled by	Type	Meaning and blank rule
26	notated_type	Composer, optional	Enum	whole, half, quarter, eighth, 16th. Blank: exporter chooses
27	notated_duration_beats	Composer, optional	Decimal	Written length if it differs. Blank: follows duration_beats
28	notated_dots	Composer, optional	Integer	Augmentation dots. Blank: exporter infers
29	tie_start	Composer, optional	Boolean	A tie begins here. Blank: no tie
30	tie_stop	Composer, optional	Boolean	A tie ends here. Blank: no tie
31	articulation	Composer, optional	Text	staccato, accent, tenuto, marcato. Blank: none
32	beam	Composer, optional	Text	begin, continue, end. Blank: exporter infers
33	midi_file_format	Policy	Integer	Standard MIDI File format; default 1
34	ticks_per_quarter	Policy	Integer	Tick resolution; default 384
35	midi_track	Mapped default	Integer	Track by staff; conductor is 0. Bach: lower 1, upper 2
36	midi_track_name	Mapped default	Text	Kept if set; otherwise from staff
37	midi_channel	Mapped default	Integer	Channel 1 to 16, independent of track. Bach: 1 and 2
38	midi_program	Mapped default	Integer	General MIDI program, zero-based; 0 = Acoustic Grand Piano. Sent only if column 39 is Yes
39	program_change_emit	Policy	Yes/No	Send a program change. Default No
40	program_source	Source	Text	Program observed, composer-set, or fallback
41	midi_bank_msb	Composer, optional	Integer	Bank select CC0, 0 to 127. Blank: no event
42	midi_bank_lsb	Composer, optional	Integer	Bank select CC32, 0 to 127. Blank: no event
43	note_off_method	Policy	Enum	note_off or note_on_velocity_zero; default note_on_velocity_zero

#	Column	Filled by	Type	Meaning and blank rule
44	note_off_velocity	Policy	Integer	Release velocity; default 0
45	absolute_tick	Computed	Integer	Onset in ticks from the start of the piece
46	onset_tick_in_measure	Computed	Integer	(onset_beat_in_measure − 1) × ticks_per_quarter
47	duration_ticks	Computed	Integer	duration_beats × ticks_per_quarter
48	end_tick	Computed	Integer	absolute_tick + duration_ticks
49	tempo_us_per_quarter	Computed	Integer	60,000,000 / tempo_bpm
50	tempo_beat_unit	Policy	Text	quarter, in every meter; other values not supported in 0.3.2
51	pitch_octave_convention	Policy	Text	C4=60
52	tick_rounding_policy	Policy	Text	nearest; every rounded value listed in the receipt
53	same_tick_event_order	Policy	Text	note_off_before_note_on
54	time_signature_num	Computed	Integer	Top number of time_signature, ignoring any grouping suffix
55	time_signature_den	Computed	Integer	Bottom number of time_signature: 4 or 8
56	key_signature_fifths	Computed	Integer	Sharps positive, flats negative; 0 for C major
57	key_signature_mode	Computed	Text	major or minor
58	controller_number	Composer, optional	Integer	Generic controller 0 to 127. Blank: no event
59	controller_value	Composer, optional	Integer	Value for column 58; 0 is a real value
60	cc1_modulation	Composer, optional	Integer	Modulation 0 to 127. Blank: no event
61	cc2_breath	Composer, optional	Integer	Breath 0 to 127. Blank: no event
62	cc7_channel_volume	Composer, optional	Integer	Channel volume; not velocity. Blank: no event
63	cc10_pan	Composer, optional	Integer	Pan; 64 is centre. Blank: no event

#	Column	Filled by	Type	Meaning and blank rule
64	cc11_expression	Composer, optional	Integer	Expression 0 to 127. Blank: no event
65	cc64_sustain	Composer, optional	Integer	Sustain pedal; 64 or more is on. Blank: no event, not pedal off
66	cc91_reverb_send	Composer, optional	Integer	Reverb send. Blank: no event
67	cc93_chorus_send	Composer, optional	Integer	Chorus send. Blank: no event
68	pitch_bend	Composer, optional	Integer	0 to 16383, centre 8192. Blank: no event
69	channel_aftertouch	Composer, optional	Integer	Channel pressure 0 to 127. Blank: no event
70	poly_aftertouch	Composer, optional	Integer	Key pressure for this note. Blank: no event
71	sysex_payload	Composer, optional	Text	Manufacturer data; not implemented. Blank: no event
72	controller_event_policy	Policy	Text	blank = emit no event
73	composition_title	Composer	Text	Title; same on every row
74	composer	Composer	Text	Composer credit; never invented
75	composition_total_measures	Computed	Integer	Number of measures in the piece
76	composition_end_tick	Computed	Integer	composition_total_measures × measure length in ticks; Bach 53,760
77	composition_duration	Computed	Text	Total length rounded down to whole seconds, Xm YYs (changed in 0.2)
78	lyric	Composer, optional	Text	Syllable for a vocal note. Blank: none
79	marker	Composer, optional	Text	Section label or cue at this onset. Blank: none
80	source_note_data	Source	Text	Origin of notes: original composition, MIDI extraction, user input
81	source_clef	Source	Text	Evidence for clef choices
82	source_key_signature	Source	Text	Evidence for key; records whether a source MIDI had a key event

#	Column	Filled by	Type	Meaning and blank rule
83	source_instrument	Source	Text	Written instrument versus emitted program
84	source_license	Source	Text	Licence of the source material
85	source_url	Source	URL	Public reference for the source

Appendix B. Copy-ready prompts

Attach this manual and the complete CSV to each prompt. For one combined prepare-and-export pass, use Appendix D.

B.1 Prepare after an editing session

> I have finished an editing session in SymphonyCSV. Following the 0.3.2 manual, validate my complete 85-column CSV and regenerate all computed values now, before export. Use deterministic code wherever available. Fill mapped defaults and policies only where they are blank. Preserve every composer input and every optional value I filled, keep existing event IDs, give new notes new IDs, and leave blank optional values blank. If inputs conflict or need musical judgment, stop and ask. Keep my original, return a new version, and give a regeneration receipt listing changes by event_id, column, old value and new value.

B.2 Refresh display values while composing

> I am still composing. Refresh the computed columns in my SymphonyCSV with deterministic code, so running_time and the other display values are current. Change nothing I entered and fill nothing optional. List what changed by event_id, column, old value and new value. This is a refresh only, not an export preparation.

B.3 Audit without changing anything

> Check my SymphonyCSV against the 0.3.2 manual without changing any value. List invalid or missing required inputs, unsupported meters, counts outside the meter's vocabulary, notes that cross a barline, stale computed columns, inconsistent whole-piece values, duplicate event IDs, blank-versus-zero errors and running_time values that break the rounding-down rule. Propose a correction plan and wait for my approval.

B.4 Start a sketch

> I am the composer. Help me draft a short original piano idea in SymphonyCSV. First ask about musical decisions that materially affect the result. Follow my tempo, meter (one of the six in section 4.1, counted with its own vocabulary), harmonic plan and note range. Enter the ten composer inputs and, where appropriate, the title and credit; leave running_time and duration_ms for calculation. I may revise the sketch freely; regenerate the whole file when I ask to prepare an export. Label your suggestions clearly and never attribute to me an idea you supplied.

B.5 Export MIDI

> If my SymphonyCSV has changed since its last pre-export regeneration, run the Appendix D preparation first. Then build a real MIDI 1.0 Standard MIDI File, Format 1. Emit only what the CSV specifies: a program change only if program_change_emit is Yes, no controller event for a blank cell. Take timing from ticks, not milliseconds. Parse the file back with a separate parser and compare every note tuple (track, channel, onset tick, pitch, velocity, duration) to the CSV. Attach the file and an export receipt. If you cannot build binary files, stop and say so.

B.6 Export MusicXML

> If my SymphonyCSV has changed since its last pre-export regeneration, run the Appendix D preparation first. Then build a MusicXML 4.0-compatible file for the supported features. Keep sounding pitches and timing exact. Use composer-set notation columns where filled; list every inferred clef, voice, beam, rest or spelling in the receipt. Validate the XML where you can, parse it back, compare sounding notes to the CSV, and attach the file and receipt. Say honestly which features are unsupported.

B.7 Make a targeted revision

> Change only the event IDs and composer inputs I specify, preserving every other choice and intentional blank. Do not run a full regeneration after this edit. Return the edited working CSV and a change list. When I ask for an export, run the Appendix D preparation first.

Appendix C. Bach reference, glossary, and sources

C.1 The reference composition

Johann Sebastian Bach, The Well-Tempered Clavier, Book I, Prelude No. 1 in C Major, BWV 846. The reference score and MIDI come from the Mutopia Project, piece 5, which lists that transcription as Public Domain. The SymphonyCSV example is a structured representation of that source, not an original work by the SymphonyCSV project.

Fact	Value
Measures	35 (16 notes in each of measures 1 to 34; a 5-note final chord)
Notes	549: 415 upper staff, 134 lower staff
Tempo and meter	60 BPM, 4/4
Resolution	384 ticks per quarter
Length	2m 20s, ending at tick 53,760

Fact	Value
Range	Upper staff D3 to A5; lower staff C2 to E4
Columns always blank	27 of 85 (notation, controller, lyric and marker columns)

C.2 Glossary

Term	Meaning
Composer inputs	The ten note fields in the first 12 columns, plus title and credit; the source of truth
Computed column	A value calculated by code from the composer inputs
Working draft	A file mid-session; its computed columns may be stale
Refresh	An optional recalculation of computed columns while composing; not an export gate
Pre-export regeneration	Validating the whole file and recalculating every dependent field before an export
Export-ready version	A new file version that has passed pre-export regeneration
Receipt	Evidence of source, output, assumptions, overwrites and checks
Simple meter	A meter whose beats divide into two, such as 2/4, 3/4, 4/4 or 5/4
Compound meter	A meter whose beats divide into three, such as 6/8
Irregular meter	A meter with unequal beats, such as 7/8 grouped 2+2+3
Count vocabulary	The fixed set of count_aloud words for one meter, each with one exact offset
Measure length (Q)	Quarter notes per measure: 4 × numerator ÷ denominator

Term	Meaning
CSV	Comma-separated values: portable, human-readable structured text
DAW	Digital audio workstation, for sequencing, recording and mixing
LLM	Large language model: composition assistant and current regeneration and export interpreter
MIDI	An event protocol and file format for musical performance data, not audio
SMF Format 1	Standard MIDI File with several time-synchronized tracks
MusicXML	An interchange format for digital sheet music
PPQ	Ticks per quarter note, the MIDI file's timing resolution
Velocity	Note-attack value, 1 to 127 for a sounding note
Voice	A notation stream for independent overlapping rhythms on one staff

C.3 Sources

- MusicXML 4.0 Final Community Group Report
- MusicXML notation tutorial
- MIDI Association: Standard MIDI Files
- MIDI Association: MIDI 1.0
- MuseScore Studio Handbook: opening and saving scores
- MuseScore Studio Handbook: working with MusicXML files
- Mutopia Project: Bach BWV 846

Appendix D. Prepare and export a SymphonyCSV

Use this prompt when an editing session is finished and you want a verified export. It runs the pre-export regeneration, validation, export and verification in one pass, with an LLM that has file and code tools.

D.1 Copy-ready prompt

SymphonyCSV: prepare and export my composition

I am providing my complete SymphonyCSV composition and the Experimental 0.3.2 manual. I have finished my current editing session. Prepare the file for export to [MIDI 1.0 Format 1 / MusicXML 4.0-compatible / both].

1. Read all 85 columns in their documented order. Do not work from a preview or truncated rows.
2. Treat my composer inputs, title, credit, explicit policies and any optional values I filled as authoritative. Do not change my music.
3. Validate notes, counts, measures, tempo, time signature, key, pitch, duration, velocity, instruments and whole-piece metadata. Check that the meter is one of the six supported, that every count is in that meter's vocabulary, that every note ends by its barline, and that event IDs are unique.
4. Recalculate every computed field from my inputs with deterministic code wherever available, taking measure length from time_signature_num and time_signature_den: MIDI pitch, beat-to-tick conversion, milliseconds, running time and end positions. Fill mapped defaults and policies only where they are blank; keep any I set. Follow the manual's rounding rules, and never derive MIDI timing from display milliseconds.
5. Keep the same 85 headers, column order, existing event IDs and intentional blanks. Give new IDs only to new notes. A blank controller means no event, not zero.
6. Keep the notation and performance details I set. Do not fill voices, beams, ties, articulations, controllers, lyrics or spellings without my approval.
7. If an input is missing, conflicting, unsupported or ambiguous, stop and ask rather than inventing a musical decision.
8. Produce a new UTF-8 85-column CSV version and leave my working file unchanged. Give a regeneration receipt: rows checked, recalculations and overwrites (event_id, column, old and new), defaults filled, assumptions and unresolved issues.
9. When that version passes validation, export the selected format. Emit MIDI events only where CSV values or documented policies call for them. Record any MusicXML notation inferences separately in the receipt.

> 10. Parse each exported file back with an independent reader. Compare every note's pitch, onset tick, duration, velocity and track or channel.
> 11. Return downloadable CSV and export files with an export receipt showing expected, matched, missing and extra notes. Never claim success without creating and parsing the files.
>
> My musical decisions remain mine. Your role is to assist, calculate, interpret the specification and verify the exports.

Revision history

Edition	Date	Changes
0.3.2	2 November 2026	Six meters (2/4, 3/4, 4/4, 5/4, 6/8, 7/8) with no schema change, requested by T and Spock; a counting vocabulary per meter; generalized timing equations; tempo in compound and irregular meters; three-tier support status; synthetic MIDI round-trip tests for all six meters
0.3.1	2 November 2026	Consistency pass: one set of terms; Appendix D fills mapped fields only when blank; refresh prompt (B.2); diagram and version references updated
0.3	2 November 2026	Session-based pre-export regeneration, set by T; ten composer inputs; Appendix D prepare-and-export prompt
0.2	2 November 2026	Regeneration and precedence rules, who-fills-what dictionary, running_time rounding rule, exporter discipline, verified 85-column Bach MIDI checkpoint; drafted with Claude
0.1	2 November 2026	First experimental manual, 85-column schema and Bach reference