



LehrPuls Exchange Format (format version 2)

This document describes in full how content gets into LehrPuls and back out again: topic trees, scenarios, all eleven question types including gap text, language versions, images, single questions without a topic tree, courses with participants, and playlists with question blocks. It is written for everyone who produces import files not in the question editor but with an AI service, a script, or from another system.

There is **one** format. What this document says about fields applies everywhere — whether the content arrives as a plain JSON file, as an `.lpx` archive, or pasted into the dialog "Create questions with AI". The only difference between these ways is images; section 2 explains it.

Contents

1. Where do I start?

2. The two wrappings: bare JSON and the `.lpx` container

3. The file header

4. Topics

5. Scenarios

6. Questions

7. The eleven question types in detail

`mc_single` — multiple choice with exactly one correct answer

`mc_multi` — multiple choice with several correct answers

`poll` — poll without a correct answer

`poll_multi` — Poll+ with multiple selection

`open` — free-text question

`numeric` — numeric value with tolerance

`estimate` — estimation question without grading

`order` — put elements into the correct order

`matching` — assign elements to categories

`info` — notice without a question

8. Gap text

9. Language versions

10. Images

11. Questions without a topic tree

12. Courses and participants

13. Playlists

14. What the import checks

15. Complete example

16. Step by step: building an `.lpx` file by hand

Checklist

1. Where do I start?

In short: hand over this document — that is all it takes.

If you want an AI service to create questions, you upload two things: this document and your own source (your lecture notes as a PDF, for example). The AI produces the content of a JSON file from them. You paste that text into LehrPuls under **Topics & Questions** in the dialog "Create questions with AI" — or you save it as a file with the extension `.json` and use **Import (.lpx, .json)**.

You do not need to select or leave out anything. An AI delivers text, and text covers this format completely — except for image files, which no AI service can supply. Questions, topics, scenarios, the English version, courses, and whole playlists all travel as plain JSON text.

LehrPuls checks the text before anything is created and reports every violation in plain words. The dialog "Create questions with AI" checks each question separately and shows all objections at once — you can hand that list back to the AI for correction. An import happens completely or not at all; a failed attempt leaves your collection untouched.

You can pass this paragraph to the AI: Produce as your result only the content of a single JSON file following this description — with no explanatory text before or after it. Follow the field names and rules exactly and do not invent additional fields. Do not use the question type `cloze`; it has its own gap syntax inside the question text and is not intended for machine-generated files. Do not reference any images.

Unknown extra fields outside `options_json` and `i18n` are silently ignored — every piece of information must therefore sit in one of the fields described here. Inside `options_json` and `i18n`, however, an unknown field causes rejection, so that a typo in a scoring field or a translation does not disappear unnoticed.

2. The two wrappings: bare JSON and the .lpx container

Both ways carry the same content in the same structure. They differ in exactly one point: **images need the container.**

	plain JSON file	.lpx container
Topics, scenarios, questions, language versions	yes	yes
Courses with participants, playlists	yes	yes
Image files	no	yes
Pasting into "Create questions with AI"	yes	no (file only)
Producible by an AI service	yes	no

An `.lpx` file is an ordinary ZIP archive with a fixed extension. Inside are exactly two things:

```
Collection.lpx  (ZIP)
└─ export.json    – all content, human-readable (UTF-8, without byte order mark)
```

```
└─ media/
   └─ 3f9ab2...c1.png  - image files named by their content hash (see section 10)
   └─ 91c07d...e4.svg
```

The plain JSON file is nothing other than the content of this `export.json`, saved on its own. Everything this document says about fields therefore describes both ways at once.

The file `export.json` must sit at the top level of the archive, otherwise LehrPuls reports "export.json missing". Images sit in the subfolder `media`; other entries in the archive are skipped. The compression method does not matter — LehrPuls itself writes `export.json` compressed and stores images uncompressed, because PNG and JPEG are already compressed.

Building the archive by hand. Create a folder holding `export.json` and the subfolder `media`. On macOS and Linux, pack it from inside that folder in the terminal with `zip -r ../Collection.lpx export.json media`. On Windows, select the file `export.json` and the folder `media` in Explorer, choose "Send to → Compressed (zipped) folder" from the context menu, and rename the result to `Collection.lpx`. Do not pack the enclosing folder — `export.json` would then sit one level too deep.

Without images, the plain JSON file is enough. Save the content of `export.json` directly as a file with the extension `.json` or `.lpx`. LehrPuls checks the content, not the extension: if the file starts with the ZIP signature bytes it is read as an archive; otherwise LehrPuls reads the first JSON object it contains. Text before and after it — an introduction, a code fence, a follow-up question from the AI — is skipped. A file without any curly brace is rejected with "neither ZIP nor JSON". LehrPuls itself writes this form as well: if you choose the file type **JSON (without images)** in the save dialog when exporting, or give the target file the extension `.json`, you get the content of `export.json` as a text file; image references stay in the text, the image files are missing.

Both files come in through the same menu entry **Import (.lpx, .json)**. The dialog "Create questions with AI" accepts pasted JSON text; for a file with images it has the **Import .lpx** button.

3. The file header

`export.json` is a single JSON object. The header consists of these fields:

Field	Required	Meaning
<code>format</code>	yes	literally <code>"lehrpuls-export"</code> — otherwise "not a LehrPuls export file"
<code>version</code>	yes	format version, currently <code>2</code> ; files of version <code>1</code> are read unchanged (section 9), a higher number is rejected as "from a newer LehrPuls version"
<code>exported_at</code>	no	timestamp of the export, display only
<code>app_version</code>	no	LehrPuls version that wrote the export, display only
<code>source_path</code>	no	topic path of the origin as a list of names; the import dialog shows it under "Origin", and when loose questions are imported without a target topic, the newly created topic is named after its last entry

Below the header come the content blocks. Each is optional, they can be combined freely, and at least one must have content — otherwise LehrPuls reports "The file contains no content":

Block	Content	Section
topics	topic tree with scenarios, questions and subtopics	4 to 10
scenarios and questions	scenarios and questions without a topic tree	11
courses	courses with participants	12
playlists	playlists with sections and question blocks	13

Unknown fields are silently skipped — with two exceptions: inside a question's `options_json` and inside `i18n` (section 9), any foreign field causes the file to be rejected, and so does a key inside `i18n` that is not a language code. All information must therefore live in the fields described here. A `scenarios` block alone does not count as content: scenarios without a topic tree only come together with `questions`.

The smallest complete file:

```
{
  "format": "lehrpuls-export",
  "version": 2,
  "topics": [
    {
      "name": "Trial topic",
      "questions": [
        {
          "qtype": "open",
          "title": "First question",
          "body_md": "What is LehrPuls?",
          "options_json": {}
        }
      ]
    }
  ]
}
```

What always applies on import. The import is all or nothing: if anything in the file is wrong, the collection stays unchanged and the message names the reason. The import always creates new entries — nothing is merged or overwritten; a topic with the same name is created a second time, and you decide yourself what to combine. Only images with identical content are not stored twice. The order in the file is the order in the collection. Imported questions carry the file name in the **Source** column.

4. Topics

A topic is a node of the tree under **Topics & Questions**. Subtopics go into `children`, nested as deep as you like:

Field	Required	Meaning
name	yes	name of the topic
i18n	no	language versions of the name, e.g. { "de": { "name" }, "en": { "name" } } — see section 9
archived	no	true creates the topic archived; default false
scenarios	no	scenarios of this topic (section 5)
questions	no	questions of this topic (section 6)
children	no	subtopics of the same structure

When importing under **Topics & Questions** you choose a **target topic** under which the tree from the file is attached; — **Top level** — creates the file's topics as new main topics. If you import the same file on the **Courses & Participants** or **Playlists** page, any topics it contains always land at the top level.

5. Scenarios

A scenario is a shared starting situation that several questions refer to — the case description before a series of tasks. In the file it sits in the topic under `scenarios`, and questions point to its key with their field `scenario`:

Field	Required	Meaning
key	yes	file-local key, any string, unique within the whole file — LehrPuls assigns <code>s1</code> , <code>s2</code> ... on export
title	yes	title, not empty
body_md	no	the scenario text (Markdown, formulas, images)
i18n	no	further language versions of title and text, e.g. { "en": { "title", "body_md" } } — see section 9
archived	no	true creates it archived

The keys have no meaning outside the file; on import they are replaced by the new database entries. A duplicate key is rejected with "Duplicate scenario key in file", a reference to an undefined key with "Scenario key is not defined in the file". The order in the file does not matter for references: all scenarios are created before all questions.

In LehrPuls, scenarios apply to their topic and all subtopics. So define a scenario in the topic that holds the questions or in one of its ancestors; technically the import also accepts references across topic boundaries, but the scenario bar of the question list then does not show the scenario as inherited. To play a question inside a playlist's question block with a scenario, point to the same key from the playlist (section 13).

6. Questions

Every question is an object in `questions` — inside a topic or, without a topic tree, at the top level (section 11):

Field	Required	Meaning
<code>qtype</code>	yes	question type, one of the eleven values in the table below
<code>title</code>	yes	short, non-empty title for the question list
<code>body_md</code>	no	the question text (Markdown, formulas, images); for gap text it carries the gaps
<code>options_json</code>	yes	JSON object with the type-specific fields; <code>{}</code> if there are none
<code>solution_md</code>	no	solution hint, shown on reveal
<code>i18n</code>	no	further language versions of the texts, e.g. <code>{ "en": { "title", "body_md", "solution_md", "options_json" } }</code> — see section 9; its <code>options_json</code> carries only the text fields at their ids
<code>difficulty</code>	no	difficulty, integer from 1 (easy) to 5 (hard)
<code>tags</code>	no	keywords, comma-separated in one string, e.g. <code>"Chapter 3, Basics"</code>
<code>time_budget_s</code>	no	planned time budget in seconds, positive integer
<code>scenario</code>	no	key of a scenario in the file (section 5)
<code>archived</code>	no	<code>true</code> creates the question archived
<code>key</code>	no	file-local question key, unique within the whole file; only needed when a playlist refers to the question (section 13)

In all Markdown fields — `body_md`, `solution_md`, the texts of answer choices, the scenario text and a playlist's work assignment — Markdown is allowed (lists, **bold**, `code`, tables), mathematical formulas in KaTeX notation between `$...$` inline or `$$...$$` on their own line; a dollar sign directly after a letter or digit (`US$`, `99$`) stays a currency sign. As everywhere in JSON, every backslash is doubled: `"$\frac{a}{b}$"` yields the formula $\frac{a}{b}$. LehrPuls corrects a single backslash before a formula name (`\Omega`) when reading. Every line break is rendered as a new line, a blank line separates paragraphs. No HTML — HTML tags are shown as literal text. Images are embedded as a Markdown image with a reference into the media folder (section 10).

The eleven question types. The rules for `options_json` are described in detail in section 7; the complete example in section 15 shows every type once. The column “Name in LehrPuls” gives the type as the interface and the sample prompts of the “Create questions with AI” dialog call it — the file always carries the value from the `qtype` column. Here is the overview:

qtype	Name in LehrPuls	Meaning	Fields in options_json
open	Open question	free-text question	max_len optional, 1 to 10000, default 500; sketch optional (true additionally allows a sketch)
mc_single	Multiple choice (one correct)	multiple choice, exactly one correct answer	choices (at least two, each with id and text_md), correct with exactly one ID
mc_multi	Multiple choice (several correct)	multiple choice, several correct answers	like mc_single , correct with at least one ID
poll	Poll	poll without grading, one vote	choices as above, correct empty or omitted
poll_multi	Poll+ (multiple selection)	Poll+ with multiple selection	like poll
numeric	Numeric	numeric value with tolerance	answer required, tolerance optional (not negative, default 0), unit optional
estimate	Estimate	estimation question without grading	unit and reference optional
order	Ordering	put into the correct order	items (at least two, each with id and text_md), correct_order with exactly all IDs
matching	Matching	assign elements to categories	categories and items (at least two each); every element names its correct category in category
info	Notice	notice without a question	url optional (http:// , https:// or mailto:), ack "none" , "ok" (default) or "yesno" , button labels ack_yes and ack_no with at most 40 characters
cloze	Fill-in-the-gaps	gap text	empty: {} — the gaps live in the question text (section 8)

IDs within a question are freely chosen, non-empty strings and must be unique; choice texts must not be empty.

7. The eleven question types in detail

mc_single — multiple choice with exactly one correct answer

```
{
  "choices": [
    { "id": "a", "text_md": "Answer A" },
    { "id": "b", "text_md": "Answer B" },
    { "id": "c", "text_md": "Answer C" }
  ],
}
```

```
"correct": ["b"]
}
```

At least two choices; the IDs are freely chosen strings (even "1" goes in quotes) but must be unique and non-empty, and the choice texts must not be empty either; `correct` contains exactly one of these IDs.

mc_multi — multiple choice with several correct answers

Same structure as `mc_single`; `correct` contains at least one ID, several are allowed.

poll — poll without a correct answer

Same structure as `mc_single`, but without grading: `correct` stays empty or is omitted. Participants pick exactly one option.

poll_multi — Poll+ with multiple selection

Like `poll`, but participants may check several options. `correct` stays empty as well.

open — free-text question

```
{ "max_len": 500 }
```

`max_len` is optional (default 500) and limits the answer length in characters (1 to 10000). `sketch` is optional: with `"sketch": true`, participants get a drawing area below the text field for pen, finger, or mouse; without it there is only the text field.

numeric — numeric value with tolerance

```
{ "answer": 42.0, "tolerance": 0.5, "unit": "m" }
```

An answer counts as correct if its distance to `answer` is at most `tolerance`. `answer` is mandatory. `tolerance` is optional (default 0 — the exact value) and must not be negative, `unit` is an optional unit for display.

estimate — estimation question without grading

```
{ "unit": "km", "reference": 384400 }
```

Both fields are optional; `reference` is the reference value shown when the question is resolved.

order — put elements into the correct order

```
{
  "items": [
    { "id": "1", "text_md": "Boil water" },
    { "id": "2", "text_md": "Grind coffee" },
  ]
}
```

```
{ "id": "3", "text_md": "Brew" }
],
"correct_order": ["1", "2", "3"]
}
```

At least two items with unique, non-empty IDs and non-empty texts; `correct_order` contains exactly all IDs in the correct order.

matching — assign elements to categories

```
{
  "categories": [
    { "id": "k1", "text_md": "Mammal" },
    { "id": "k2", "text_md": "Bird" }
  ],
  "items": [
    { "id": "i1", "text_md": "Dolphin", "category": "k1" },
    { "id": "i2", "text_md": "Penguin", "category": "k2" }
  ]
}
```

At least two categories and two items, IDs unique and non-empty in each list, texts non-empty; `category` names the correct category of the item and is the grading.

info — notice without a question

```
{ "ack": "ok" }
```

Shows the `body_md` as an announcement. `ack` is optional (default `"ok"`) and controls the participants' acknowledgment: `"none"` (read only), `"ok"` (confirm with one button) or `"yesno"` (choose between two buttons); with `"yesno"`, the optional fields `ack_yes` and `ack_no` label the two buttons, with `"ok"` only `ack_yes` may label the single button, with `"none"` both stay empty; `ack_no` is only allowed together with `ack_yes`, and a label has at most 40 characters. Optionally `url` carries a link, which must start with `http://` or `https://` and must not contain spaces — or an email address as `mailto:name@university.example` (with `?subject=...` if desired).

The eleventh type, `cloze` (fill-in-the-gaps), uses its own gap syntax inside the question text and therefore has a section of its own (8). It is not intended for AI-generated files — an AI service leaves this type out.

8. Gap text

For the question type `cloze`, `options_json` stays empty and the gaps, including their solutions, sit directly in the question text: `{{solution}}` marks a gap, a vertical bar separates equivalent alternatives — `{{Vienna|Wien}}` accepts both. At least one gap must be in the text, and every gap needs at least one non-empty solution. Gaps also work inside table cells and lists. There is no notation for showing `{{` literally.

```
{
  "qtype": "cloze",
  "title": "Capitals",
  "body_md": "The capital of Austria is {{Vienna|Wien}}, that of France {{Paris}}.\nWater boils at {{100}} °C.",
  "options_json": {}
}
```

When comparing answers, LehrPuls trims surrounding whitespace, collapses repeated spaces and ignores case; if solution and input are both numbers, the numeric value counts and a comma is treated as a decimal point (22 , 22.0 and 22,0 are equal). The width of the input field follows the longest solution. The solutions never leave the lecturer's computer: participants receive neutral placeholders until the question is revealed.

9. Language versions

A session runs in one language throughout. That is why not only the questions carry several language versions, but also the texts around them — the scenario text before a series of questions, the work assignment of a question block, and the names of sections and topics.

Four kinds of object have a field `i18n` for this. It is built the same way everywhere: an object with the language code as its key, holding the same field names as the main fields. Any code of two or three lowercase letters counts as a language code, optionally with a suffix — `de` , `en` , `fr` , `pt-br` . All languages rank equally; which of them a teacher maintains is set in the Cockpit under **Settings** → **Content languages**.

Object	Fields per language
Question (section 6)	<code>title</code> , <code>body_md</code> , <code>solution_md</code> , <code>options_json</code>
Scenario (section 5)	<code>title</code> , <code>body_md</code>
Playlist node (section 13)	<code>name</code> , <code>preamble_md</code>
Topic (section 4)	<code>name</code>

The main fields (`title` , `body_md` , `name` ...) carry the version of the first available language. They are the fallback: if a text is missing in the requested language, it is shown from there — half-translated collections stay playable. For files you write yourself, a simple rule applies:

- If the main fields are in **German**, there is nothing more to do. LehrPuls adopts them as the German version on import; further languages go into `i18n` . This is how files of format version 1 were built, and they are still read this way.
- If the main fields are in **another language** — a purely English collection, for instance — repeat the same texts in `i18n` under their code (`"en"`). This tells LehrPuls that there is no German version.

Export files written by LehrPuls always carry all versions in `i18n` , including `de` .

For questions, the `options_json` of a language version holds **only the text fields**, anchored to the ids in the question's `options_json`: `choices` , `items` , and `categories` as lists of { `"id"` , `"text_md"` } , plus `unit` , `ack_yes` , and `ack_no` where the type knows them. An id that does not exist in the question's

`options_json` causes the file to be rejected. All other fields do not belong in a language version, because they apply to all languages: the scoring fields (`correct`, `correct_order`, `answer`, `tolerance`, `reference`, `category`) as well as the settings `max_len`, `url`, and `ack`. A file that copies them there is rejected. If a question type has no text fields in `options_json` — such as the free-text question `open` or the fill-in-the-gaps question `cloze` —, the language version leaves out `options_json` entirely; the English version of a free-text question is therefore just `{ "title", "body_md", "solution_md" }`. For fill-in-the-gaps questions, every version must have the same number of gaps in the same order: the first gap of the English text corresponds to the first gap of the German text. Each gap is scored against the solutions of all versions — "sun" counts just like "Sonne".

```
{
  "qtype": "mc_single",
  "title": "Ort der Fotosynthese",
  "body_md": "In welchem Zellorganell findet die Fotosynthese statt?",
  "options_json": {
    "choices": [
      { "id": "a", "text_md": "Chloroplast" },
      { "id": "b", "text_md": "Mitochondrium" }
    ],
    "correct": ["a"]
  },
  "solution_md": "Die Fotosynthese läuft in den Chloroplasten ab.",
  "i18n": {
    "en": {
      "title": "Site of photosynthesis",
      "body_md": "In which organelle does photosynthesis take place?",
      "options_json": {
        "choices": [
          { "id": "a", "text_md": "Chloroplast" },
          { "id": "b", "text_md": "Mitochondrion" }
        ]
      },
      "solution_md": "Photosynthesis takes place in the chloroplasts."
    }
  }
}
```

For a scenario it is only the title and the text:

```
{
  "key": "s1",
  "title": "Werksplanung",
  "body_md": "Ein Werk wird geplant ...",
  "i18n": { "en": { "title": "Plant planning", "body_md": "A plant is being planned ..." } }
}
```

For a playlist node it is the node's name and its work assignment:

```
{
  "name": "Kostenblock",
  "kind": "complex",
  "preamble_md": "Bearbeiten Sie die Fragen in Zweiergruppen.",
  "i18n": { "en": { "name": "Cost block", "preamble_md": "Work on the questions in pairs." } }
}
```

When a session runs in English, the scenario text, the work assignment, and the block name appear in that language, just like the questions. A text without an English version stays as it is — the session does not break off, it shows the first available language at that point.

The teacher chooses the question language when starting a session; participants can switch between the available languages on their own device. Answers and statistics are anchored to the ids, not to the text — they are the same for all languages.

Not translatable are course and participant names: they appear only in administration, never during play. The name of the playlist root also stays single-language — it is an administrative name; what gets translated are the sections and blocks below it. Topic names never appear during play either; their language versions serve only the work in the Cockpit, whose topic tree follows the content language.

10. Images

Images in LehrPuls are content-addressed: the file name is the SHA-256 hash of the file content as 64 lowercase hexadecimal characters, followed by the extension. In Markdown an image is referenced by exactly this name, and in the archive it lies under the same name in the folder `media`:

```
![Circuit diagram]
(media/9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08.png)
```

The permitted extensions are `png`, `jpg`, `gif`, `webp` and `svg`; JPEG files must be named `jpg`, not `jpeg`. A file may be at most 10 MB. On import, LehrPuls recomputes the hash of every image file and compares it with the name — if it does not match, the whole import is aborted with "Image file does not match its name" before anything reaches the database. An image already present in the target collection is not stored a second time; an image referenced several times belongs in the archive only once.

Computing the hash. On a Mac and on Linux, `shasum -a 256 image.png` prints the value; on Windows, `certutil -hashfile image.png SHA256` or, in PowerShell, `Get-FileHash image.png`. Rename the file to `<hash>.png` and put it under `media/`. A script that generates the file computes the hash while copying.

Where images may appear. In every Markdown field: in the question text, in the solution hint, in the texts of answer choices, in the scenario text and in a playlist's work assignment. The export scans all these fields for references and packs the files it finds.

Image size. A size level may follow the reference, without quotation marks: `![Image](media/...png small)`. The levels are `small`, `medium`, `large`, `extra large` and `original` (German `klein`, `mittel`, `groß`, `sehr groß`); without a level, `large` applies. The projector sizes by height, participant devices and print by width.

Missing files. A reference without a matching file is not an error: the question is created and the image stays empty. The same holds for bare JSON, which never carries images. The import preview lists such references as missing images. The reference stays valid — as soon as the file reaches the media folder by another route, for example by importing an `.lpx` file that contains the same image, it appears.

11. Questions without a topic tree

Single questions may sit in the file without a surrounding topic — this is how LehrPuls exports selected questions from the question list. They then live in the block `questions` at the top level, with the scenarios they reference in the block `scenarios` next to it:

```
{
  "format": "lehrpuls-export",
  "version": 2,
  "source_path": ["Medicine", "Internal"],
  "scenarios": [
    { "key": "s1", "title": "Case Meier", "body_md": "A 54-year-old patient presents
with ..." }
  ],
  "questions": [
    {
      "qtype": "open",
      "title": "Initial diagnosis",
      "body_md": "What is your suspected diagnosis?",
      "options_json": { "max_len": 500, "sketch": true },
      "scenario": "s1"
    }
  ]
}
```

When importing under **Topics & Questions**, these questions and scenarios land in the chosen target topic. If no target is chosen, LehrPuls creates a new main topic for them — named after the last entry of `source_path`, or else after the file name without extension; for pasted text without a file it is called "Import".

12. Courses and participants

A course is exported as its setup: name, semester, description, purge date and participant list — without sessions and answers.

Field	Required	Meaning
name	yes	course name, not empty
semester	no	semester as text, e.g. "Fall 2026"
description	no	description
purge_after	no	purge date as YYYY-MM-DD
archived	no	true creates the course archived; default false
participants	no	the participant list

Every entry of the participant list:

Field	Required	Meaning
display_name	yes	display name, not empty and unique within the course
import_ref	no	student number or other identifier
email	no	e-mail address
note	no	note
active	no	false sets the entry inactive; default true
short_code	no	the person's code for switching devices: one capital letter other than I and 0, followed by three digits, e.g. "K417"; if it is missing, LehrPuls assigns one

```
{
  "format": "lehrpuls-export",
  "version": 2,
  "courses": [
    {
      "name": "Fundamentals of Electrical Engineering 1",
      "semester": "Fall 2026",
      "description": "Compulsory course, first semester",
      "purge_after": "2027-09-30",
      "participants": [
        {
          "display_name": "Anna Sample",
          "import_ref": "12345678",
          "email": "anna.sample@university.example",
          "short_code": "K417"
        },
        { "display_name": "Ben Example", "note": "Repeating", "active": false }
      ]
    }
  ]
}
```

The import runs from the **Courses & Participants** page; if a file contains courses alongside topics or playlists, they are imported on every page and announced in the preview. Such a file contains personal data and leaves LehrPuls's protected storage — pass it on only with a legal basis. An archived course keeps its archive state: LehrPuls then writes `"archived": true`, and the import creates it archived again. Older LehrPuls versions ignore the field and create it as an ordinary course.

A person's code travels along: LehrPuls writes it as `short_code` on export, and the import adopts it, so that switching devices by code keeps working after a course has moved. If a value does not have the form of a code or occurs a second time in the same course, that is not an error — the person then gets a new code, just as when the field is missing. In files you write yourself, it is best to leave the field out. Older LehrPuls versions (up to 1.14.0) ignore the field and assign new codes.

13. Playlists

A playlist is a tree: the root, sections below it, and within them question entries and question blocks. In the file, the entries refer to questions in the same export via file-local keys — in the block `topics` or `questions`, each with the field `key` set. This keeps the file self-contained; it carries no course or semester relation.

Field	Required	Meaning
<code>name</code>	yes	name of the node, not empty
<code>kind</code>	no	<code>"section"</code> for a section (default) or <code>"complex"</code> for a question block; the topmost node is always created as the root, whatever is written here
<code>preamble_md</code>	no	the work assignment of a question block (Markdown, formulas, images)
<code>i18n</code>	no	further language versions of name and work assignment, e.g. <code>{ "en": { "name", "preamble_md" } }</code> — see section 9
<code>scenario</code>	no	key of the scenario the question block shows as its preamble
<code>skipped</code>	no	<code>true</code> marks the node as skipped
<code>archived</code>	no	<code>true</code> creates the playlist archived; applies only to the topmost node, default <code>false</code>
<code>items</code>	no	the question entries of this node, in order
<code>children</code>	no	child nodes of the same structure
<code>after_items</code>	no	child nodes only: this many question entries of the parent node come before this child node; if the field is missing, it follows all questions

Questions and child nodes of a node can alternate, for example an exercise section first and then the individual questions. `after_items` gives the position of a child node: `0` places it before the first question, `2` after the second. Several child nodes with the same value follow in their order in `children`. Older LehrPuls versions ignore the field and place all child nodes after the questions.

A question entry has three fields: `question` with the key of the question (required), `peer_instruction` (optional, `true` turns peer instruction on; meaningful only for `mc_single` and `mc_multi`) and `priority` (optional, `1` = must, `2` = should, `3` = could; if the field is missing or the value is invalid, “should” applies). A question block holds its sub-questions as `items`; question blocks inside question blocks are not intended. A

reference to a key that no question in the file carries is rejected with "Question key is not defined in the file", a key assigned twice with "Duplicate question key in file".

```
{
  "format": "lehrpuls-export",
  "version": 2,
  "topics": [
    {
      "name": "Museum",
      "scenarios": [
        {
          "key": "s1",
          "title": "Special exhibition",
          "body_md": "The city museum is planning a special exhibition for the fall ..."
        }
      ],
      "questions": [
        {
          "key": "q1",
          "qtype": "open",
          "title": "Concept",
          "body_md": "Outline the exhibition concept.",
          "options_json": { "max_len": 800 },
          "scenario": "s1",
          "time_budget_s": 600
        },
        {
          "key": "q2",
          "qtype": "open",
          "title": "Budget",
          "body_md": "Draw up a rough cost plan.",
          "options_json": { "max_len": 800 },
          "scenario": "s1",
          "time_budget_s": 600
        },
        {
          "key": "q3",
          "qtype": "mc_single",
          "title": "Warm-up",
          "body_md": "How many museums does the city have?",
          "options_json": {
            "choices": [ { "id": "a", "text_md": "3" }, { "id": "b", "text_md": "7" } ],
            "correct": ["b"]
          }
        }
      ]
    }
  ],
  "playlists": [
    {
      "name": "Project week",
```

```

    "children": [
      {
        "name": "Warm-up",
        "items": [ { "question": "q3", "peer_instruction": true } ]
      },
      {
        "name": "Station 1 · Concept and budget",
        "kind": "complex",
        "scenario": "s1",
        "preamble_md": "Work on the tasks from the museum management's point of
view.",
        "items": [ { "question": "q1" }, { "question": "q2" } ]
      }
    ]
  }
}

```

The import first creates the topics, scenarios and questions and then the playlist with references to the new questions. The playlist appears at the top of the list on the **Playlists** page. If you import the file under **Topics & Questions**, the playlists come along just the same; the topic tree then lands in the chosen target topic. An archived playlist keeps its archive state: LehrPuls then writes `"archived": true` on the topmost node, and the import creates it archived again. Older LehrPuls versions ignore the field and create it as an ordinary playlist.

14. What the import checks

Before creating anything, the import dialog shows the origin (`source_path`) and the scope of the file — topics, questions, scenarios, images, courses, participants, playlists and question entries. During creation the file is validated as foreign input; the most common messages and their causes:

Message	Cause
This is not a LehrPuls export file (neither ZIP nor JSON)	the file does not start with the ZIP signature bytes and contains no <code>{</code> anywhere — plain text or an empty file, for example
This is not a LehrPuls export file (export.json missing)	<code>export.json</code> is not at the top level of the archive
This is not a LehrPuls export file	<code>format</code> is not <code>"lehrpuls-export"</code>
Export file error: ... at line ..., column ... (...)	JSON syntax error ("not valid JSON"), missing required field or wrong data type ("unexpected content"); the message names line and column, and the parentheses hold the reader's wording, such as <code>missing field</code> with the name of the missing field
The file contains no content	all four content blocks are empty
Unknown question type	<code>qtype</code> is none of the eleven values
Invalid options: unknown field ...	<code>options_json</code> contains a field the question type does not know
i18n.en.options_json: unknown field „max_len“	a language version's <code>options_json</code> contains a field that is not a text field — <code>max_len</code> , <code>url</code> , <code>ack</code> , and the scoring fields belong only in the question's <code>options_json</code> (section 9)
Mark exactly one correct choice, Provide at least two choices ...	a rule of the question type is violated (see section 6 and section 7)
Write at least one gap {{...}} into the question text	gap text without a gap
Choices need a text or an image	an answer choice has no non-empty <code>text_md</code>
Scenario key is not defined in the file	<code>scenario</code> points to a key that no scenario in the file carries
Question key is not defined in the file	a playlist entry points to a <code>key</code> that no question carries
Image file does not match its name	the hash of the file in the archive does not match its file name
Unsupported image type	the extension is not <code>png</code> , <code>jpg</code> , <code>gif</code> , <code>webp</code> or <code>svg</code>
"..." is already on the participant list	two participants with the same display name in the same course

For bare JSON in the dialog "Create questions with AI", LehrPuls checks every question individually and shows all objections at once; the file import stops at the first one. In both cases the collection stays unchanged.

15. Complete example

An archive `Physics.lpx` with one image:

```

Physics.lpx
├─ export.json
└─ media/
    └─ 9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08.png

```

The matching `export.json` shows every question type once, a scenario with an image, a subtopic, a playlist with a question block, and a course:

```

{
  "format": "lehrpuls-export",
  "version": 2,
  "exported_at": "2026-09-15T09:00:00Z",
  "source_path": ["Physics"],
  "topics": [
    {
      "name": "Physics",
      "scenarios": [
        {
          "key": "s1",
          "title": "Pendulum setup",
          "body_md": "A simple pendulum of length  $l = 1.0\text{ m}$  is displaced
by 5°.\n\n![[Setup]]
(media/9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08.png medium)"
        }
      ],
      "questions": [
        {
          "key": "q1",
          "qtype": "mc_single",
          "title": "Period",
          "body_md": "What does the period of the pendulum depend on for small
displacements?",
          "options_json": {
            "choices": [
              { "id": "a", "text_md": "the mass" },
              { "id": "b", "text_md": "the string length" },
              { "id": "c", "text_md": "the displacement" }
            ],
            "correct": ["b"]
          },
          "solution_md": " $T = 2\pi\sqrt{l/g}$  – the mass does not appear.",
          "difficulty": 2,
          "tags": "Mechanics, Oscillations",
          "time_budget_s": 60,
          "scenario": "s1"
        },
        {
          "key": "q2",
          "qtype": "numeric",
          "title": "Compute the period",

```

```

    "body_md": "Compute the period in seconds ( $g = 9.81 \text{ m/s}^2$ ).",
    "options_json": { "answer": 2.01, "tolerance": 0.05, "unit": "s" },
    "solution_md": "$T = 2\pi\sqrt{1.0/9.81} \approx 2.01 \text{ s}$",
    "difficulty": 3,
    "time_budget_s": 120,
    "scenario": "s1"
  },
  {
    "key": "q3",
    "qtype": "mc_multi",
    "title": "Forms of energy",
    "body_md": "Which forms of energy does the pendulum convert into one
another?",
    "options_json": {
      "choices": [
        { "id": "a", "text_md": "kinetic energy" },
        { "id": "b", "text_md": "potential energy" },
        { "id": "c", "text_md": "chemical energy" }
      ],
      "correct": ["a", "b"]
    },
    "difficulty": 2
  },
  {
    "key": "q4",
    "qtype": "poll",
    "title": "Prior knowledge",
    "body_md": "Did you take physics in high school?",
    "options_json": {
      "choices": [ { "id": "a", "text_md": "Yes" }, { "id": "b", "text_md": "No" }
    ]
  },
  {
    "key": "q5",
    "qtype": "poll_multi",
    "title": "Interests",
    "body_md": "Which topics interest you? (multiple selection)",
    "options_json": {
      "choices": [
        { "id": "a", "text_md": "Mechanics" },
        { "id": "b", "text_md": "Optics" },
        { "id": "c", "text_md": "Thermodynamics" }
      ]
    }
  },
  {
    "key": "q6",
    "qtype": "open",
    "title": "Sources of error",
    "body_md": "Name two sources of error when measuring the period."
  }
]

```

```

    "options_json": { "max_len": 400 },
    "solution_md": "Reaction time when stopping the watch, air resistance, too
large a displacement.",
    "time_budget_s": 180
  },
  {
    "key": "q7",
    "qtype": "estimate",
    "title": "Fall time",
    "body_md": "Estimate: how long does a stone fall from a height of 20 m?",
    "options_json": { "unit": "s", "reference": 2.02 }
  },
  {
    "key": "q8",
    "qtype": "order",
    "title": "Measurement procedure",
    "body_md": "Put the steps into the correct order.",
    "options_json": {
      "items": [
        { "id": "1", "text_md": "Displace the pendulum" },
        { "id": "2", "text_md": "Time ten oscillations" },
        { "id": "3", "text_md": "Divide the time by ten" }
      ],
      "correct_order": ["1", "2", "3"]
    }
  },
  {
    "key": "q9",
    "qtype": "matching",
    "title": "Quantities and units",
    "body_md": "Assign each quantity its unit.",
    "options_json": {
      "categories": [ { "id": "k1", "text_md": "second" }, { "id": "k2",
"text_md": "meter" } ],
      "items": [
        { "id": "i1", "text_md": "period", "category": "k1" },
        { "id": "i2", "text_md": "string length", "category": "k2" }
      ]
    }
  },
  {
    "key": "q10",
    "qtype": "cloze",
    "title": "Complete the formula",
    "body_md": "The period grows with the {{square root|root}} of the string
length; doubling the length increases it by a factor of {{1.41|1,41|√2}}.",
    "options_json": {}
  },
  {
    "key": "q11",
    "qtype": "info",

```

```

        "title": "Lab instructions",
        "body_md": "You will find the complete lab instructions in the lecture notes,
chapter 4.",
        "options_json": { "url": "https://university.example/physics/notes.pdf",
"ack": "yesno", "ack_yes": "Read", "ack_no": "Later" }
    }
],
"children": [
    {
        "name": "Exercises",
        "questions": [
            {
                "qtype": "numeric",
                "title": "Pendulum on the Moon",
                "body_md": "What period would the same pendulum have on the Moon ( $g = 1.62\text{ m/s}^2$ )?",
                "options_json": { "answer": 4.94, "tolerance": 0.1, "unit": "s" },
                "difficulty": 4,
                "scenario": "s1"
            }
        ]
    }
]
}
],
"playlists": [
    {
        "name": "Lecture 3 · Oscillations",
        "children": [
            {
                "name": "Warm-up",
                "items": [ { "question": "q4" }, { "question": "q11" } ]
            },
            {
                "name": "Pendulum experiment",
                "kind": "complex",
                "scenario": "s1",
                "preamble_md": "Evaluate the experiment in pairs.",
                "items": [ { "question": "q1", "peer_instruction": true, "priority": 1 }, {
"question": "q2" }, { "question": "q6", "priority": 3 } ]
            }
        ]
    }
]
},
"courses": [
    {
        "name": "Physics 1",
        "semester": "Fall 2026",
        "participants": [ { "display_name": "Anna Sample" }, { "display_name": "Ben
Example" } ]
    }
]

```

```
]
}
```

16. Step by step: building an .lpx file by hand

1. Create a folder containing the file `export.json` and the subfolder `media`.
2. Write `export.json` following sections 3 to 13 — easiest by copying the complete example from section 15 and trimming it. Save as UTF-8 without a byte order mark.
3. For every image, compute the SHA-256 hash, rename the file to `<hash>.<extension>` and put it under `media/`; reference it in Markdown as `media/<hash>.<extension>`.
4. Pack `export.json` and `media` — not the folder above them — into a ZIP archive and give it the extension `.lpx`.
5. Import the file into LehrPuls via **Import (.lpx, .json)** and check in the preview that scope and origin are right. If the import fails, the message names the reason (section 14); the database stays unchanged.
6. The quickest way to learn the format is a real file: export a small topic with one image from LehrPuls, rename the `.lpx` file to `.zip` and look inside.

Checklist

- `format` is `"lehrpuls-export"`, `version` is `2`; at least one of the blocks `topics`, `questions`, `courses`, `playlists` is filled.
- Every question has `qtype`, `title` and an `options_json` matching its type without foreign fields; gap texts have at least one gap in the text.
- Scenario keys and question keys are unique within the file, and every reference points to a defined key.
- Every image lies in the archive as `media/<sha256>.<extension>`, the hash matches, the extension is `png`, `jpg`, `gif`, `webp` or `svg`.
- `export.json` sits at the top level of the archive and is UTF-8 without a byte order mark.
- Courses have a name, participants a display name unique within the course; pass on files with participant lists only with a legal basis.

Additionally for machine-generated files: Does the result really contain nothing but the JSON text, with no introduction, closing remark, or code fence? Are `format` and `version` set literally? Does every question carry a non-empty `title` and an `options_json` (`{}` if need be)? Is the question type one of the eleven allowed values, and has `cloze` been avoided? Are all ids within a question unique and non-empty? Does `options_json` inside `i18n` contain only text fields (`choices`, `items`, `categories`, `unit`, `ack_yes`, `ack_no`) — and is it left out there entirely for free-text questions, so no `max_len`? Is there no image reference in the text?