kanban-board

Kanban project boards in LaTeX

Abstract

The kanban-board package typesets compact project boards composed of named columns and task cards. A card may carry a priority, an author, a deadline, and a completion marker. Deadlines can be expressed either as calendar dates or as months relative to a project start date. Two rendering layouts are provided: a fixed horizontal board suited to short summaries and a page-breakable layout for longer task lists. Named palettes and light or dark board modes can be selected at package, document, or board level, and a single card may carry a tinted background. Deadline chips distinguish overdue, current, and future values. A reference date can be fixed for reproducible output. Otherwise the compilation date is used. Five common numeric date formats are available, with US notation as the initial default. This manual describes the public interface, gives complete usage examples, and explains the date and configuration models.

Andreas Willibald Weber

Version 1.0.0 (2026/11/15)

Contents

1 Introduction **1**

2 Requirements and installation **1**

3 The board environment **2**

3.1 Board options . . . 3

3.2 A first board . . . 4

4 Configuration hierarchy **4**

4.1 Default column lists . . . 6

5 Declaring tasks **6**

5.1 Task options . . . 8

6 Deadline semantics **9**

6.1 Calendar dates . . . 9

6.2 Date formats . . . 10

6.3 Project months . . . 11

6.4 Reproducible reference dates . . . 12

6.5 Worked deadline example . . . 12

7 Color and layout model **13**

7.1 Selecting a palette . . . 13

7.2 Column geometry . . . 16

7.3 Light and dark modes . . . 17

7.4 Card background tints . . . 18

7.5 Labels and public style hooks . . . 19

8 Diagnostics and limitations **20**

9 Further usage examples **24**

9.1 Reusable document setup . . . 24

9.2 Selecting a date format . . . 26

9.3 Light and dark boards in one document . . . 27

9.4 Changing the gap between columns . . . 28

9.5 A page-breakable board 29
9.6 A fixed board on a landscape page 32

10 License and maintenance **35**

11 Implementation **35**
11.1 Public command contracts 35
11.2 Configuration and data collection 36
11.3 Rendering strategies 36
11.4 Metadata and validation 37
11.5 Bootstrap state 37
11.6 Palette and column registries 42
11.7 Deadline engine 50
11.8 Persistent configuration 61
11.9 Card components and renderers 67
11.10 Task collection 72
11.11 Board lifecycle 74

1 Introduction

A Kanban board represents work by assigning cards to named stages such as "Backlog", "In progress", and "Done". The package follows that model directly. A board uses either the built-in columns, the current document default, or a board-local list selected with the `columns` environment option. The author then assigns each task to exactly one of those columns. Columns are rendered in declaration order, and tasks retain their order within each column.

The package is intended for reports, project notes, teaching material, and other documents in which a board should be generated from ordinary LaTeX source. It is not an interactive project-management system: tasks are local document data, and the package neither persists nor automatically moves them.

At a glance

Core environment `\begin{kanban}[options]`, with board-local columns selected by the `columns={...}` option when required.

Task command `\kanbantask[options]{column}{text}`, with `\kbtask` as a concise alias when that short name is available.

Shared setup `\kanbansetup{key=value,...}`, with `\kbset` as a concise alias when that short name is available.

Visual choices Named palettes, light or dark mode, tinted card backgrounds, and fixed or page-breakable layouts.

Date choices US, English, German, ISO, and slash-separated year-first display formats, with validated Gregorian classification.

Column geometry Synchronized headings and transparent column bodies in the fixed layout, with configurable separation.

2 Requirements and installation

The package requires a LaTeX format dated 2020-10-01 or later and loads xcolor, etoolbox, tikz, tcolorbox, and paracol. On formats older than 2021-06-01, whose option processor cannot parse brace groups in package option lists, kvoptions is loaded additionally. These dependencies are included in the principal TeX distributions. Continuous integration runs the regression and documentation builds on both a current TeX Live installation and an actual TeX Live 2020 installation. The declared format boundary is also exercised locally through the official latexrelease compatibility mechanism. The regression suite is run with pdfTeX, LuaTeX, and XeTeX.

To install from the documented source, run

```
latex kanban-board.ins
```

and copy `kanban-board.sty` to a directory searched by TeX. For a private installation, that directory is normally a suitable location below `TEXMFHOME`. No manual installation is necessary when the package is installed through a TeX distribution.

Load the package in the preamble. No options are required:

```
\usepackage{kanban-board}
```

Initial document defaults may also be set while loading the package:

```
\usepackage[
  columns={Planned,Active,Complete},
  palette=muted,
  mode=light,
  date-format=us,
  startdate=07/01/2026,
  reference-date=2026-07-18,
  columnsep=2mm
]{kanban-board}
```

These defaults can be changed later with `\kanbansetup` (or its concise alias `\kbset`) or overridden by an individual board. The following sections explain this hierarchy in detail.

3 The board environment

`kanban` (*env.*) The environment has the syntax

`\begin{kanban}[`⟨*board options*⟩`]`
⟨*task declarations*⟩
`\end{kanban}`

If `columns` is omitted, or if `columns={}` is supplied, the board uses the current document-wide column list. The package initially defines that list as `Backlog,Doing,Done`. A non-empty comma-separated value replaces the default for this board only. The additional braces around the value protect its commas from the surrounding key list. Each name is trimmed. Its spelling and its formatting are kept for the visible heading. Its lookup key is the purified text of the name in uppercase. Ordinary macros are expanded, formatting commands and braces are removed, and letter case is ignored. Task targets undergo the same conversion, so `ColName`, `colname`, and `COLNAME` all identify the same column. Likewise, `\emph{Soon}` and `Soon` denote the same column, and so do `Q{1}` and `Q1`. Column names must be unique after normalization and must not contain commas.

Each board is independent. When a board begins, the package first resolves the applicable column list. It then creates an empty task list and a zero count for every

resolved column before collecting task declarations. The visible board is produced only when the environment ends. This deferred construction is why tasks can be written in any cross-column order while still appearing beneath the correct headings.

3.1 Board options

The optional argument accepts the following keys.

`break` Selects the page-breakable layout. Each board column becomes a paracol column with an independent colored header followed by its cards. This layout is appropriate when task lists may continue onto a later page. Headers are printed at the start of the board and are not repeated on continuation pages.

`columns={`⟨***column list***⟩`}` Replaces the current document-wide column list for this board only. The value must be enclosed in braces because it contains commas. An empty value retains the current document default.

`columnsep=`⟨***length***⟩ Sets the horizontal distance between adjacent columns for this board. The default is `2mm`. This value overrides the document default for the current board only and is used by both the fixed and page-breakable layouts. The accepted forms of the value are described in the subsection on column geometry.

`palette=`⟨***name***⟩ Selects a previously defined color palette for this board. The local selection does not change the palette used by later boards.

`date-format=us`, `english`, `german`, `iso`, **or** `ymd` Selects the numeric format used to print validated calendar dates on this board. The initial default is `us`. This setting also resolves dates whose slash-separated day and month are both at most 12. A local selection does not change the format used by later boards.

`startdate=`⟨***date***⟩ Sets the project start date for this board. It accepts any supported calendar syntax and is used only when interpreting project-month deadlines such as `M2`. This value overrides the document default for the current board only.

`reference-date=`⟨***date***⟩ Sets the date against which calendar and project-month deadlines are classified for this board. Without this key, the package uses the TeX compilation date. The exact alias `today` is also accepted. An explicit reference date makes archival and automated builds reproducible.

`mode=light` **or** `mode=dark` Selects the visual mode for this board. The default is `light`. A local mode does not change the mode of subsequent boards.

`labels={`⟨***label options***⟩`}` Replaces one or more semantic chip labels for this board. The accepted nested keys are `high`, `medium`, `low`, and `done`. Omitted labels retain their current document-wide values.

The `break` key is board-local by design. It is not accepted as a package option or by `\kanbansetup`. Without `break`, the package constructs a two-row tcolorbox raster inside one full-width minipage. The first row contains the synchronized headings and the second contains the synchronized column bodies. The complete board is therefore one horizontal object and does not break naturally across pages. This default is preferable for compact boards. Task cards occupy the complete available

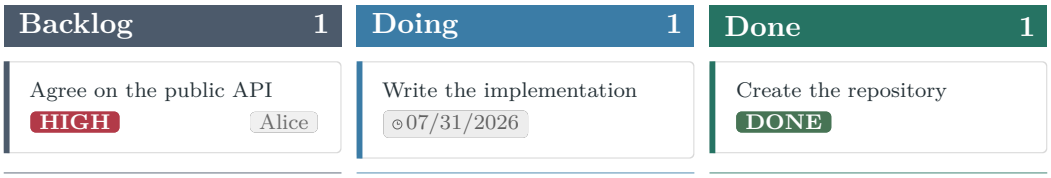
width of the column body in both layouts.

3.2 A first board

The following complete example applies the environment syntax and default fixed layout described above. Because no column list is supplied, it uses the built-in `Backlog`, `Doing`, and `Done` stages and assigns one task to each stage. The deliberately varied capitalization of the task targets also demonstrates case-insensitive assignment.

```
\begin{kanban}
  \kanbantask[prio=high,author=Alice]
    {backlog}{Agree on the public API}
  \kanbantask[due=07/31/2026]
    {DoInG}{Write the implementation}
  \kanbantask[completed]
    {DONE}{Create the repository}
\end{kanban}
```

The example produces the following board:



4 Configuration hierarchy

The package uses a consistent hierarchy for column lists, palettes, visual modes, date formats, project start dates, reference dates, semantic labels, and column separation. This makes document-wide settings easy to reuse while keeping local exceptions explicit.

1. Package options establish the initial defaults.
2. `\kanbansetup` changes the defaults for boards that follow it.
3. Environment options override the defaults for one board only, including the board-local `columns` key.

The most local value has priority. For example:

```
\usepackage[
  columns={Planned,Active,Complete},
  palette=muted,
  mode=light,
  date-format=us,
  reference-date=2026-07-18,
  columnsep=2mm
]{kanban-board}

% Uses every package default, including the configured columns.
\begin{kanban}
  ...
\end{kanban}

% Changes defaults for subsequent boards.
\kbset{
  columns={Queued,Processing,Stored},
  palette=monochrome,
  mode=dark,
  date-format=iso,
  reference-date=2026-07-18,
  columnsep=1.5mm
}

% Uses the revised document defaults.
\begin{kanban}
  ...
\end{kanban}

% Local columns, default palette, light mode, English dates, 3 mm gap.
\begin{kanban}[
  columns={Todo,Doing,Done},
  palette=default,mode=light,date-format=english,columnsep=3mm
]
  ...
\end{kanban}

% Uses Queued, Processing, and Stored again.
\begin{kanban}
  ...
\end{kanban}
```

`\kanbansetup` The setup command accepts a comma-separated key list:

> `\kanbansetup{columns={`⟨*column list*⟩`},palette=`⟨*name*⟩`,`
> `mode=`⟨*mode*⟩`,date-format=`⟨*format*⟩`,`
> `startdate=`⟨*date*⟩`,reference-date=`⟨*date*⟩`,`
> `columnsep=`⟨*length*⟩`,labels={`⟨*label options*⟩`}}`

When the name is available, the shorter `\kbset` command is an exact alias. The longer name is the canonical form because it is less likely to conflict with another package. If `\kbset` already exists when kanban-board is loaded, the package preserves the existing definition, issues a warning, and leaves `\kanbansetup` fully available. A key may be omitted when its current value should remain unchanged. Setup calls

affect later boards. They do not redraw boards that have already been typeset. The `columns` value must be enclosed in braces because it is itself a comma-separated list. An empty `columns` value is rejected with a warning, and the previous document default remains active.

4.1 Default column lists

With no configuration, every board that omits its `columns` option uses the three columns `Backlog`, `Doing`, and `Done`. This provides a useful conventional board without requiring repeated declarations:

```
\begin{kanban}
  \kanbantask{backlog}{Record the request}
  \kanbantask{Doing}{Prepare the response}
  \kanbantask{dOnE}{Archive the decision}
\end{kanban}
```

A document can establish different defaults when the package is loaded:

```
\usepackage[columns={Planned,Active,Complete}]{kanban-board}
```

or change them from a particular point onward:

```
\kbset{columns={Queued,Processing,Stored}}
```

In both cases, later environments may omit the `columns` key. Supplying a board-local key remains the appropriate way to describe an exceptional board:

```
\begin{kanban}[columns={Requested,Approved,Rejected}]
  ...
\end{kanban}
```

This option is local. It does not change the default used by the next environment. Letter case and formatting need not match, because declarations and task targets share one purified uppercase lookup key. The heading shows the spelling and the formatting used in the column list. Other differences remain significant. For example, `In review` and `Review` are distinct identifiers. A list such as `columns={Q{1},Q1}` repeats the key `Q1`. If a list repeats a name after normalization, the package reports an error, retains the first occurrence, and ignores the repeated occurrence. This recovery permits the remaining unique columns to be rendered if compilation is continued.

5 Declaring tasks

`\kanbantask` A task declaration has the syntax

`\kanbantask[`⟨*task options*⟩`]{`⟨*column*⟩`}{`⟨*text*⟩`}`

When the name is available, `\kbtask` is an exact alias with the same signature and behavior. The longer `\kanbantask` name is the canonical interface because it is less likely to conflict with another package. If `\kbtask` already exists when kanban-board is loaded, the package preserves the existing definition, issues a warning, and leaves `\kanbantask` fully available. The column must have been declared by the enclosing board. The target is purified and converted to uppercase before lookup in the same way as a declared column name. The assignment therefore ignores surrounding spaces, letter case, formatting commands, and braces. The task text is ordinary LaTeX material and may contain emphasis, citations, robust commands, and other commands suitable for a `tcolorbox`. It may also contain paragraph breaks, written as `\par` or as a blank line. They are rendered as separate paragraphs in the card. The option list and the column argument do not accept a paragraph break. A blank line inside either of them produces the LaTeX error "Paragraph ended before `\kanbantask` was complete". LaTeX then discards the declaration up to that point and reads the remainder as ordinary text. Fragile or unusually complex material should be tested in the intended document class.

The column argument may not be empty. A missing assignment or a normalized name that is absent from the board produces a package error. The diagnostic identifies the invalid input, lists the valid columns by their purified uppercase keys, and explains how to correct the declaration. The invalid task is not counted or rendered, which leaves the remaining board data structurally consistent if compilation is continued after the error. A task declared outside a `kanban` environment similarly produces a direct error instead of modifying residual board data.

For example, both tasks below are assigned to the single displayed column `Research`:

```
\begin{kanban}[columns={Research,Review}]
  \kanbantask{research}{Collect the evidence}
  \kanbantask{ReSeArCh}{Document the method}
\end{kanban}
```

By contrast, the following fragments are invalid:

```
\kanbantask{}{A task without a target}
\kanbantask{Archive}{A task for an undeclared stage}
```

For a board containing `RESEARCH` and `REVIEW`, the corresponding errors state that the first task has no assigned column and that `Archive` is unknown. Both diagnostics also report `RESEARCH,REVIEW` as the valid column set.

Formatting in a column name is ignored for lookup in the same way as letter case. The following board declares an emphasized heading and addresses it with plain text. The heading shows the emphasized text, and an error for this board would report `SOON,LATER` as the valid column set:

```
\begin{kanban}[columns={\emph{Soon},Later}]
  \kanbantask{soon}{Prepare the next release}
  \kanbantask{\emph{Soon}}{Collect the release notes}
\end{kanban}
```

Both tasks are assigned to the first column.

The package deliberately does not define the generic `\task` alias. That name is used by several document classes and packages. Leaving it untouched prevents load-order-dependent conflicts. The namespaced `\kbtask` form provides a concise alternative without claiming the generic name.

5.1 Task options

`prio=high` Adds a red high-priority pill, labelled HIGH by default.

`prio=medium` **or** `prio=med` Adds an orange medium-priority pill, labelled MED by default.

`prio=low` Adds a green low-priority pill, labelled LOW by default.

`author=`⟨***text***⟩ Places a neutral author chip at the right of the metadata row.

`deadline=`⟨***value***⟩ Adds a deadline chip beneath the metadata row. The alias `due` has exactly the same meaning.

`late` Forces an existing deadline chip to use the overdue style. The option has no visible effect unless `deadline` or `due` is also present.

`completed` Adds a green marker, labelled DONE by default. Completion is presentational: it does not move the task or suppress its deadline.

`bg=`⟨***color***⟩ Tints the surface of the card with the given xcolor color or mixing expression. The alias `background` has exactly the same meaning. Section 7.4 describes the result in both modes.

Task options are local to one declaration. An unknown option name generates a warning that names the key:

```
Unknown task option 'colour' on line 12. The option is
ignored.
```

Priority values are trimmed and compared without regard to letter case. An unrecognized value produces a warning, and the priority chip is omitted. A background value must name a color known to xcolor. An unknown value produces a warning, and the card keeps the default surface.

Several options may be combined in one declaration:

```
\kanbantask[
  prio=high,
  author=Alice,
  due=07/31/2026
]{Review}{Approve the release candidate}
```

The package does not infer workflow state from metadata. In particular, `completed` adds a marker but does not move the task to a column named "Done". The column argument remains the only source of task placement.

Task text may contain ordinary inline markup:

```
\kanbantask{Doing}{
  Compare \emph{pdfLaTeX} and \emph{LuaLaTeX} output
}
```

A card may also contain more than one paragraph. A blank line or `\par` in the text argument starts a new paragraph inside the card:

```
\kanbantask{Doing}{
  Compare the output of both engines.

  Record the differences in the project log.
}
```

Authors should keep card text concise. Long paragraphs are valid, but they make narrow boards difficult to scan.

6 Deadline semantics

Deadline classification is performed when a card is rendered. By default, the comparison day is obtained from the TeX primitives `\year`, `\month`, and `\day`. Consequently, recompiling the same source on a later date may change a deadline chip from neutral to current or overdue. The `reference-date` key replaces that moving comparison day with a validated, fixed date when reproducible output is required. The syntax of a deadline is checked when the task is declared, so a warning about an invalid value names the line of that declaration.

6.1 Calendar dates

A deadline is interpreted as a calendar date when its complete expanded value has one of the supported numeric forms. Ordinary macros are expanded before the check, so a macro that expands to a date is classified. Robust and protected commands are not expanded. A value such as `\emph{soon}` or `{\textbf{2026-07-18}}` is therefore a descriptive label. It is printed with its formatting and without automatic classification. The parser recognizes month-first and day-first dates with a final year, German dot notation, ISO notation, and slash-separated year-first notation.

For example:

```
\newcommand\releasedate{2026-08-15}
\kanbantask[due=07/31/2026]{Review}{Approve the release}
\kanbantask[due=\releasedate]{Planned}{Archive the release}
\kanbantask[due=\emph{soon}]{Planned}{Announce the release}
```

The first two deadlines are calendar dates. The third is a label. A date before the active reference date is red, a date equal to it is orange, and a future date is neutral. When no reference date has been configured, the compilation date is the active reference. Every notation requires a four-digit year from 0001 to 9999. Month and day may be written with one or two digits. Dates are checked against the Gregorian month lengths, including the leap-year rule. A value that has date syntax but does not denote a valid date produces a warning and is retained as a neutral label. The same applies to a value that has the shape of a date but a year of fewer than four digits. The input `18.07.26` is therefore reported rather than read as a date in the year 26. Other descriptive values, including labels beginning with `M` or containing punctuation, are displayed without automatic classification. The `late` option may be used when such a label must appear overdue.

6.2 Date formats

The `date-format` key selects how every validated calendar date is printed. The package provides five common numeric conventions:

Key value	Convention	18 July 2026
`us`	month/day/year	`07/18/2026`
`english`	day/month/year	`18/07/2026`
`german`	day.month.year	`18.07.2026`
`iso`	year-month-day	`2026-07-18`
`ymd`	year/month/day	`2026/07/18`

`us` is the initial default. Output components are zero-padded. Month and day may be entered with one or two digits, so `2026-7-8` and `2026-07-08` denote the same date and are both printed as `07/08/2026` under the default. Selecting a format follows the same three-level hierarchy as the other board settings:

```
\usepackage[date-format=english]{kanban-board} % initial document default
\kbset{date-format=german}                     % subsequent boards

\begin{kanban}[date-format=iso]                % this board only
  ...
\end{kanban}
```

The compact spelling `dateformat` is accepted as an exact key alias, although `date-format` is the canonical spelling used throughout this manual. Format values are trimmed and compared without regard to letter case. The canonical values in the table should be preferred in new documents. For compatibility, `american` maps

to `us`. The values `uk` and `british` map to `english`. The values `de` and `deutsch` map to `german`, and `year-first` maps to `ymd`.

Calendar parsing is intentionally more permissive than display formatting: an unambiguous date in any supported notation is accepted and normalized to the selected output convention. A year-last slash date can be ambiguous. For example, `07/08/2026` denotes either 8 July or 7 August. The `english` and `german` selections interpret such a value as day/month/year. The `us`, `iso`, and `ymd` selections use month/day/year for this otherwise ambiguous input. If only one ordering denotes a valid Gregorian date, that ordering is accepted regardless of the current selection. ISO input is consequently the safest form for task data exchanged between documents, because its component order is unambiguous. Project-month values such as `M2` and descriptive labels are never reformatted.

The `startdate` and `reference-date` settings are normalized to an internal ISO representation as soon as their containing key list has been processed. Their values are therefore immutable snapshots: a later macro redefinition or board-local display-format change cannot reinterpret an inherited date. Within one package, setup, or board option list, `date-format` is resolved before either stored date is normalized, irrespective of the textual order of those keys. Task deadlines remain board data and are interpreted under the board's selected format when their cards are rendered.

6.3 Project months

A complete value of the form `M`$\langle n \rangle$, where $\langle n \rangle$ is an integer from 1 to 999999, denotes project month $\langle n \rangle$. Leading zeros are permitted, so `M01` and `M0000001` both denote month 1. The bound of six digits applies to the significant digits. It keeps malformed input outside TeX's integer arithmetic. A value outside this range, such as `M0`, `M000`, or `M1000000`, produces a warning and remains a neutral label. The initial project start date may be supplied as a package option:

```
\usepackage[startdate=07/15/2026]{kanban-board}
```

It may be changed for subsequent boards through the common setup interface:

```
\kbset{startdate=2026-07-15}
```

Finally, a board can override the default locally:

```
\begin{kanban}[columns={Planned,Active,Done},startdate=07/15/2026]
\end{kanban}
```

The precedence order is therefore board option, current `\kbset` default, and package option. A board-local value does not alter subsequent boards. An empty setup value clears the persistent start date for later boards. An empty board value disables an inherited start date for that board only:

```
\kbset{startdate={}}
\begin{kanban}[startdate={}]
  ...
\end{kanban}
```

Accepted start dates are stored canonically rather than as their original input tokens. For example, a US `07/08/2026` start date remains 8 July even if a later board selects `date-format=english` for its visible deadlines. This separation prevents a display preference from changing project-month boundaries.

Project months begin on the day number of the start date. Thus, for a project beginning on 15 July, `M1` runs from 15 July until 14 August and `M2` begins on 15 August. If that day does not exist in a later calendar month, the boundary is clamped to the final valid day of that month. A project beginning on 31 January therefore enters `M2` on 28 February, or on 29 February in a leap year. Its following boundary is 31 March. This rule prevents a short month from extending the preceding project month unintentionally.

A project month earlier than the current one is overdue. The current project month is orange, and a later month is neutral. If no start date is available, the value is displayed without automatic classification. An invalid start date produces a package warning and does not replace the previously valid value.

6.4 Reproducible reference dates

The optional `reference-date` setting fixes the comparison day used by both calendar deadlines and project months. It follows the same package, setup, and board-local hierarchy as `startdate`:

```
\usepackage[reference-date=2026-07-18]{kanban-board}
\kbset{reference-date=2026-08-01}

\begin{kanban}[reference-date=2026-08-15]
  ...
\end{kanban}
```

The key alias `today` has exactly the same meaning. An empty setup value restores the compilation date for later boards, while an empty board value restores it for that board only. Like a project start date, an accepted reference date is validated, expanded, and stored canonically. Invalid input produces a warning and preserves the preceding value. This facility is useful for archived reports, regression tests, and automated publication systems in which a rebuild should not alter status colors.

6.5 Worked deadline example

The following board uses both public key names and displays all three visual states with calendar dates. The manual fixes its reference date as 18 July 2026, so the first date lies in the past, the second equals the reference date, and the third lies in

the future.

```
\begin{kanban}[columns={Overdue,Due now,Upcoming}]
  \kanbantask[deadline=01/01/2000]
    {Overdue}{Resolve the review comments}
  \kanbantask[due=07/18/2026]
    {Due now}{Complete the current milestone}
  \kanbantask[deadline=2099-12-31]
    {Upcoming}{Prepare the next milestone}
\end{kanban}
```

The red chip denotes a past date, the orange chip denotes the active reference date, and the neutral chip denotes a future date. The first and third values also demonstrate US and ISO calendar input. Both are printed in the selected format, which is US notation here because the board uses the default. The keys `deadline` and `due` are exact aliases. Their choice does not determine the color. The fixed reference date keeps the rendered result stable whenever the manual is compiled.



A complete version of this demonstration, in both light and dark mode, is provided as `kanban-board-deadlines-example.tex`.

7 Color and layout model

By default, columns draw from an eight-color palette in declaration order: slate, blue, teal, sage, amber, clay, rose, and plum. A selected custom palette may contain a different number of colors. When the number of columns exceeds the number of entries, the palette repeats. Priority and status colors are independent of the column color.

In the default layout, the nominal width of each column is calculated from the available line width and the number of columns. Large boards can therefore become too narrow for useful text. In practice, three or four columns fit comfortably in portrait documents. Wider boards often benefit from landscape orientation, a larger paper size, or the `break` layout.

7.1 Selecting a palette

Version 1.0.0 provides named palettes. The initial palette may be selected as a package option:

```
\usepackage[palette=muted]{kanban-board}
```

The built-in names are `default`, `muted`, and `monochrome`. The `default` palette contains the original eight-color sequence.

`\kbset{palette=`⟨*name*⟩`}` changes the default for subsequent boards:

```
\kbset{palette=monochrome}
```

A board may override that default without changing later boards:

```
\begin{kanban}[columns={Todo,Doing,Done},palette=default]
  ...
\end{kanban}
```

`\kanbandefinepalette` A new palette is registered with

`\kanbandefinepalette{`⟨*name*⟩`}{`⟨*color list*⟩`}`

where the second argument is a non-empty, comma-separated list of colors or xcolor expressions. For example:

```
\definecolor{universityBlue}{HTML}{164A7B}
\definecolor{universityGold}{HTML}{A67C00}
\kbdefinepalette{university}{
  universityBlue,
  teal!70!black,
  universityGold
}
\kbset{palette=university}
```

The unstarred form uses white heading text for every entry. If a palette contains both dark and light backgrounds, the starred form records an explicit foreground color with each background:

`\kanbandefinepalette*{`⟨*name*⟩`}{` ⟨*background*⟩`/`⟨*foreground*⟩`,...}`

For example:

```
\kbdefinepalette*{accessible}{
  facultyNavy/white,
  facultyGold/kanbanInk,
  facultyBerry/white
}
```

The slash separates the two xcolor expressions. Every entry in a starred definition must contain exactly one slash and two non-empty values. Otherwise the complete definition is rejected with a warning. This explicit pairing permits suitable heading

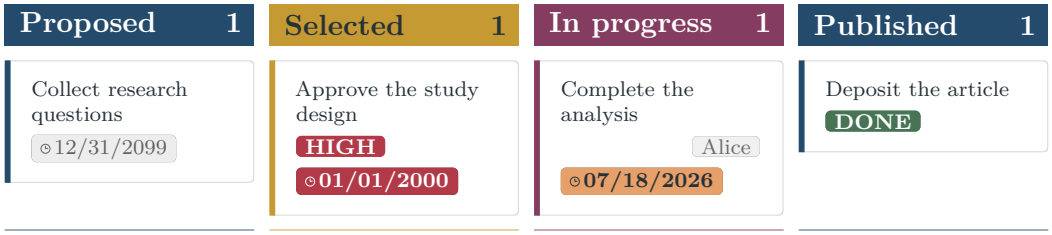
contrast for arbitrary custom colors. It does not attempt to calculate or certify a contrast ratio, because xcolor expressions may depend on the surrounding document's color model and named-color definitions. When its name is available, the concise command `\kbdefinepalette` is an exact alias. If another package has already defined that name, kanban-board preserves it, reports a warning, and requires the canonical `\kanbandefinepalette` command. Surrounding spaces in a palette name are ignored, but palette names are case-sensitive. Palette registration is document-wide: the definition command creates or replaces a named registry entry, while package, setup, and board `palette` keys only select an existing entry. Defining an existing name replaces its stored color list. If that name is the current document default, later boards immediately use the replacement. The name does not need to be selected again. A board takes a local snapshot of the palette and its foreground list when it begins, so redefining the name while that board is being collected cannot alter the board already in progress. An unknown or empty palette produces a warning and leaves the current selection unchanged.

A palette may contain any positive number of colors. When a board has more columns than the palette has entries, selection wraps cyclically. A three-color palette is therefore applied as colors 1, 2, 3, 1, 2, 3, and so forth. The built-in colors are dark enough to give white heading text a contrast ratio of at least 4.5:1. Authors of custom palettes remain responsible for preserving adequate contrast. The starred interface makes the foreground choice explicit for every palette entry. The following example defines a three-color institutional palette with dark text on its lighter gold background. Since the board has four columns, its fourth heading reuses the first palette entry.

```
\definecolor{facultyNavy}{HTML}{244A6C}
\definecolor{facultyGold}{HTML}{C69A32}
\definecolor{facultyBerry}{HTML}{843D60}
\kbdefinepalette*{faculty}{
  facultyNavy/white,
  facultyGold/kanbanInk,
  facultyBerry/white
}

\begin{kanban}[
  columns={Proposed,Selected,In progress,Published},
  palette=faculty
]
  \kanbantask[deadline=12/31/2099]
    {Proposed}{Collect research questions}
  \kanbantask[prio=high,deadline=01/01/2000]
    {Selected}{Approve the study design}
  \kanbantask[author=Alice,
    due=07/18/2026]
    {In progress}{Complete the analysis}
  \kanbantask[completed]{Published}{Deposit the article}
\end{kanban}
```

It produces the following board:



The complete compilable source is `kanban-board-palette-example.tex`.

7.2 Column geometry

The `columnsep` key controls the distance between adjacent columns. Its default value is `2mm`. The key accepts three forms of value. The first form is a non-negative explicit dimension such as `2mm` or `0.5em`. The number and the unit may be separated by spaces, as in `2 mm`, and a physical unit may carry the `true` prefix, as in `2truemm` or `2 true mm`. The second form is a single LaTeX length register such as `\linewidth` or a length declared with `\newlength`. The third form is a single TeX dimension or glue parameter such as `\parindent`, `\baselineskip`, `\hsize`, or `\parskip`. A `\let` copy of such a parameter is accepted as well. A macro that expands to one of these forms is also accepted. Every other value is rejected with a warning, and the current value is kept. Rejected values include `\dimexpr` expressions, scaled registers such as `0.5\linewidth`, the `true` prefix on a font-relative unit as in `2trueem`, math glue such as `\thinmuskip`, a bare number, a bare unit, and robust commands such as `\emph{2mm}`. The check takes place before TeX's dimension scanner is entered, so a rejected value does not produce secondary "missing number" or "illegal unit" errors. The help text of the warning reads "Use a non-negative explicit dimension such as 2mm, a LaTeX length register, or a TeX length parameter such as `\hsize`." An initial value may be set as a package option:

```
\usepackage[columnsep=3mm]{kanban-board}
```

It may be changed for subsequent boards with the shared setup command:

```
\kbset{columnsep=1.5mm}
```

A single board may override the current default without affecting later boards:

```
\begin{kanban}[columns={Todo,Doing,Done},columnsep=4mm]
  ...
\end{kanban}
```

The same value is applied to the fixed layout and to the column separation used by the `break` layout. In the fixed layout the package subtracts the total separation from the available line width before dividing the remaining width among the columns. Each task card then occupies the complete available width of its column body.

The value must also be feasible for the resolved number of columns. For $\langle n \rangle$ columns, the product of $\langle n \rangle$ minus one and `columnsep` must be smaller than `\linewidth`. If

the total gap is equal to or wider than the available line width, the package reports an error and uses `0pt` for that board. This recovery value preserves positive column width while leaving the document-wide default unchanged.

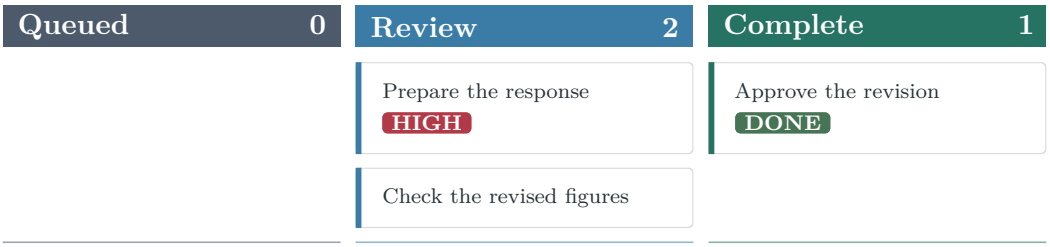
The fixed layout is constructed as a two-row raster. All heading cells use the height of the tallest heading, including headings that wrap onto more than one line. All column bodies use the height of the body containing the most material, so empty or short columns extend to the same bottom edge. The column body itself is transparent in both visual modes. Only its bottom rule remains visible, while the left and right rules are omitted. Task cards keep their light or dark background, border, and colored accent.

Equal-height measurements are stored in the document's auxiliary file. Authors should therefore compile the document twice after changing a column heading or task list. In the page-breakable layout, heading heights are also synchronized at the beginning of the board, but column bodies remain independent so that they can continue naturally across pages. The initial headings are not repeated on continuation pages.

The following example uses concise headings and includes an empty column. The shorter labels are preferable in narrow boards because they keep the workflow stages visually compact. Longer headings remain supported and are equalized automatically when wrapping is unavoidable.

```
\begin{kanban}[columns={Queued,Review,Complete}]
  \kanbantask[prio=high]{Review}
    {Prepare the response}
  \kanbantask{Review}
    {Check the revised figures}
  \kanbantask[completed]{Complete}{Approve the revision}
\end{kanban}
```

It produces a board whose headings and bottom edges remain aligned:



7.3 Light and dark modes

The default `light` mode uses dark text on white cards, neutral gray metadata chips, and white headings on the selected palette colors. The `dark` mode uses light text on dark cards. Author chips and neutral `deadline` or `due` chips deliberately retain their light-mode colors in both modes, giving metadata a stable visual identity. Automatic status colors remain semantic: overdue values are red and values due now are amber in either mode. In both modes the area behind the cards is transparent, so the surrounding page remains visible in the vertical space between cards and at

the foot of a synchronized column. Palette colors continue to identify columns in both modes.

The initial mode may be selected as a package option:

```
\usepackage[mode=dark]{kanban-board}
```

It can be changed for subsequent boards through the common setup interface:

```
\kbset{mode=dark}
```

A board may then override that default locally:

```
\begin{kanban}[columns={Todo,Doing,Done},mode=light]
  ...
\end{kanban}
```

Mode selection affects only the board. It deliberately does not call `\pagecolor` and does not alter the surrounding document text. A dark board can therefore be embedded safely in an otherwise light document. Unknown mode names produce a warning and leave the current selection unchanged. A robust command in the value, as in `mode=\emph{dark}`, is reported as unknown in the same way.

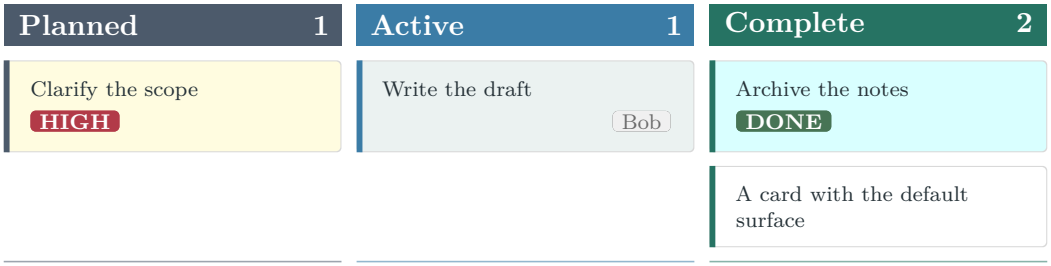
The examples section below compares the two modes with identical column geometry and explicit deadline states. A separate custom-palette example is provided as `kanban-board-palette-example.tex`.

7.4 Card background tints

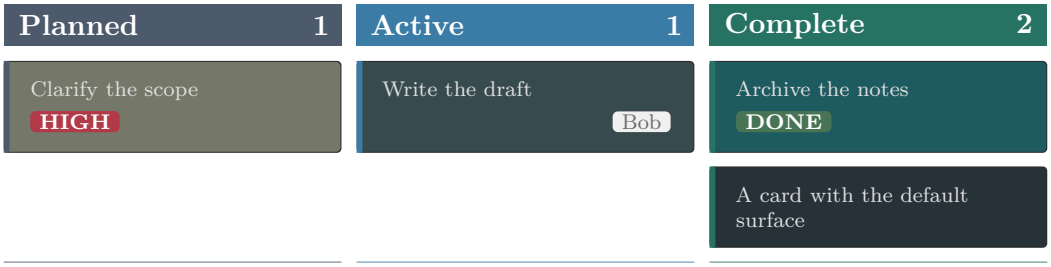
The surface of an individual card can carry a touch of color. The task option `bg` accepts an xcolor color name or a mixing expression of the form ⟨*name*⟩`!`⟨*percentage*⟩`!`⟨*name*⟩, optionally negated with a leading minus sign. The alias `background` is accepted as well. In light mode the surface is white with 15 percent of the color. In dark mode the surface is the dark card color with 20 percent of the color. The larger share keeps the tint visible on a dark surface, and the text keeps a contrast ratio above 4.5:1 with the built-in text colors for any tint. Heading, column accent, metadata chips, and text colors are unchanged, so a tint groups cards visually without competing with the semantic colors.

```
\begin{kanban}[columns={Planned,Active,Complete}]
  \kanbantask[bg=yellow,prio=high]{Planned}{Clarify the scope}
  \kanbantask[bg=kanbanTeal!60,author=Bob]{Active}{Write the draft}
  \kanbantask[background=-red,completed]{Complete}{Archive the notes}
  \kanbantask{Complete}{A card with the default surface}
\end{kanban}
```

It produces the following board:



The same declarations on a dark board mix the tints into the dark surface:



The value is checked when the task is declared. Every name in the expression must be known to xcolor at that point, and the expression must follow the mixing grammar above. Any other value, including a robust command, produces a warning that names the line of the task, and the card keeps the default surface:

```
Unknown color 'nosuchcolor' on line 12. The card keeps the
default background.
```

A tint is a property of one card. There is no board-level or document-level default, because a tint marks an exception rather than a theme. The mixing percentages are fixed in this release.

7.5 Labels and public style hooks

The visible semantic labels can be adapted without redefining internal commands. The `labels` key accepts the nested keys `high`, `medium`, `low`, and `done`. It follows the same package, setup, and board-local precedence as the other configuration keys:

```
\usepackage[
  labels={high=URGENT,medium=NORMAL,low=ROUTINE,done=FINISHED}
]{kanban-board}

\kbset{labels={done=APPROVED}}

\begin{kanban}[labels={high=IMMEDIATE}]
  ...
\end{kanban}
```

A board takes a snapshot of all four labels when it begins. An omitted label keeps its current value. An empty value is rejected with a warning. Label text is ordinary

short LaTeX material, although concise uppercase terms are normally easiest to read inside the small chips.

Seven namespaced tcolorbox styles provide a controlled extension point for typographic and spacing changes:

> `kanban/header`, `kanban/card`, `kanban/column-body`, `kanban/priority-chip`, `kanban/author-chip`, `kanban/deadline-chip`, and `kanban/completed-chip`.

The styles are initially empty and are applied after the structural package defaults. For example:

```
\tcbset{
  kanban/header/.append style={fontupper=\bfseries\sffamily},
  kanban/card/.append style={arc=1mm,outer arc=1mm},
  kanban/author-chip/.append style={left=1.5mm,right=1.5mm}
}
```

These hooks are document-wide tcolorbox settings. The header, card, and chip styles apply to both rendering strategies. `kanban/column-body` applies only to the fixed renderer because the page-breakable renderer leaves each paracol stream without an enclosing body box. Authors should preserve `width=\linewidth` for cards and should test changes to padding in narrow columns. The named colors `kanbanHigh`, `kanbanMed`, and `kanbanLow` remain available to advanced users, but changing either foreground or background colors also transfers responsibility for readable contrast to the document author. In the defaults, the orange medium and due-now chips use dark ink. The red and green semantic chips use white. Together with the revised built-in palettes, these pairs meet a 4.5:1 reference contrast ratio for ordinary text.

8 Diagnostics and limitations

The package distinguishes warnings from errors. A warning reports input that can be ignored safely. The package then keeps the current value or omits the affected element. An error reports a declaration that cannot be interpreted unambiguously. Errors use the LaTeX2e package error mechanism. Every warning is a named message of the l3msg module `kanban-board`.

Warnings. Every warning follows one pattern. The first sentence names the subject, quotes the offending value, and states the line of the input that caused the warning. The second sentence states how the package recovers. An unknown setup, board, task, label, or board label option names the key and is ignored:

```
Unknown setup option 'colour' on line 3. The option is
ignored.
Unknown board label option 'urgent' on line 9. The option
is ignored.
```

An unknown palette, mode, or date format keeps the current selection. The board-local form names the scope, for example “Unknown board palette” or “Unknown board mode”:

```
Unknown palette 'brand' on line 4. The current palette is
kept.
Unknown mode 'blue' on line 5. The current value is kept.
Unknown date format 'dmy' on line 6. The current value is
kept.
```

A robust command in such a value, as in `mode=\emph{dark}`, is reported as unknown in the same way. A palette definition is rejected as a whole when its name is empty, when its color list is empty, or when a starred definition contains a malformed pair. The starred form reports “has no color/text pairs” for an empty list:

```
Empty palette name on line 7. The definition is ignored.
Palette 'empty' on line 8 has no colors. The definition is
ignored.
Palette 'broken' on line 9 has a malformed color/text pair.
The definition is ignored.
```

An empty column list, an empty label, an invalid start or reference date, and an invalid column separation keep the current value:

```
Empty column list on line 10. The current list is kept.
Empty label 'high' on line 11. The current value is kept.
Invalid startdate '31.02.2026' on line 12. The current
value is kept.
Invalid columnsep value '2' on line 13. The current value
is kept.
```

The board-local forms of these messages read “Empty board label”, “Invalid board startdate”, “Invalid board reference-date”, and “Invalid board columnsep value”. A malformed or negative column separation is intercepted before TeX emits a low-level number or unit error. An unknown priority omits the chip. An unknown card background keeps the default surface. A deadline that resembles a calendar date or a project month but lies outside the validated range is displayed as a neutral label:

```
Unknown priority 'urgent' on line 14. The priority chip is
omitted.
Unknown color 'nosuchcolor' on line 15. The card keeps the
default background.
Invalid deadline date '18.07.26' on line 16. The card
shows a neutral label.
Invalid project-month deadline 'M0' on line 17. The card
shows a neutral label.
```

Deadline warnings are issued when the task is declared, so the line is that of the declaration. Descriptive labels, including a deadline that contains a robust command, remain neutral without a diagnostic.

A diagnostic raised by a package option says "in the package options" instead of naming a line, because the package file is being read at that moment:

```
Invalid startdate '31.02.2026' in the package options. The
current value is kept.
```

The optional short commands `\kbtask`, `\kbset`, and `\kbdefinepalette` are installed only when their names are unused. A pre-existing definition is never overwritten, and the canonical long command remains available in every case. The warning is issued while the package loads and therefore carries no line context:

```
\kbset is already defined. Use \kanbansetup instead.
```

The same pattern applies to `\kbtask` and `\kbdefinepalette`.

Message identifiers. Every message belongs to the l3msg module `kanban-board`. The following table lists the identifier of each message and the subject of its text.

Identifier	Subject
`unknown-key`	Unknown setup, board, task, or label option
`unknown-value`	Unknown mode or date format
`unknown-palette`	Unknown palette
`empty-palette-name`	Empty palette name
`empty-palette`	Palette without colors or color/text pairs
`malformed-palette-pair`	Malformed color/text pair
`empty-column-list`	Empty column list
`empty-label`	Empty label
`invalid-date-setting`	Invalid start date or reference date
`invalid-dimension`	Invalid column separation
`unknown-priority`	Unknown priority
`unknown-color`	Unknown card background
`invalid-deadline`	Invalid deadline date
`invalid-project-month`	Invalid project-month deadline
`alias-taken`	Short command already defined

An individual message can be silenced or redirected with `\msg_redirect_name:nnn`. The following preamble lines turn the unknown-key warning off and demote the unknown-value warning to an entry in the log file:

```
\ExplSyntaxOn
\msg_redirect_name:nnn {kanban-board}{unknown-key}{none}
\msg_redirect_name:nnn {kanban-board}{unknown-value}{info}
\ExplSyntaxOff
```

A redirection is local to the current TeX group. Placed in the preamble, it applies to the whole document.

Errors. A task with an empty column argument, a task outside a board, a task assigned to a column that is absent from the board, or a column list containing a duplicate after normalization generates a package error with corrective help. The error for a task lists the valid columns by their purified uppercase keys, so a column declared as `\emph{Soon}` appears as `SOON`. Duplicate names recover by keeping the first occurrence and discarding later occurrences. Invalid tasks are neither counted nor rendered. Nested `kanban` environments are also rejected because each board owns one active collection context. The nested environment produces one error, and task declarations inside it are then ignored without a cascade of secondary "outside a board" errors. A non-negative column separation whose combined gaps leave no column width produces an error and is replaced by `0pt` for the affected board. A paragraph break inside the option list or the column argument of a task produces the LaTeX error "Paragraph ended before `\kanbantask` was complete". LaTeX then discards the declaration up to that point and reads the remainder as ordinary text.

Constraints. Authors should also observe the following constraints.

- The resolved default or explicit list must contain at least one column. The built-in default satisfies this requirement, and an empty value passed to the `columns` setup key is rejected.
- Column lookup keys are the purified uppercase text of the names. Ordinary macros are expanded, and formatting commands and braces are removed before the comparison. Headings show the spelling and the formatting of the column list. Names must not contain commas and must remain unique after normalization.
- Mode and priority values are trimmed and compared without regard to letter case. Date-format values and their documented aliases follow the same normalization rule. A robust command in any of these values is reported as unknown. Palette names remain case-sensitive.
- A card background must be a color name known to xcolor or a mixing expression of the form ⟨*name*⟩`!`⟨*number*⟩`!`⟨*name*⟩ whose names are all known. Any other value is reported and ignored.
- Column separation must be a non-negative explicit dimension, a single LaTeX length register, or a single TeX length parameter, and the sum of all inter-column gaps must be smaller than the available line width.
- Calendar dates, project start dates, and reference dates may use US or

English slash notation with a final year, German dot notation, ISO hyphen notation, or slash-separated year-first notation. The year always has four digits. Ambiguous year-last slash dates follow the selected date format. Calendar dates are validated for years from 0001 to 9999 and are normalized for display.

- Only complete calendar dates and values `M1` through `M999999` are classified automatically. Leading zeros in the month number are permitted. All other deadline values, including values that contain a robust command, are treated as labels and remain neutral unless `late` is present. The `M` in a project-month value is uppercase and case-sensitive.
- Task text is passed as a command argument, so verbatim material cannot be used inside a card. Paragraph breaks are permitted in the text argument but not in the option list or the column argument.
- The fixed raster layout cannot split an individual board across pages. Use `break` for long boards.
- The `break` layout prints synchronized headings once at the beginning of the board. It does not repeat them on continuation pages because the supported paracol interface provides independent column streams, not a repeating per-column heading row.
- The `break` layout should not be used inside a pdflscape `landscape` environment. Use the fixed layout for a landscape board and ensure that the complete board fits on that page.
- Task counts include every successfully assigned task, irrespective of its completion status.

9 Further usage examples

The distribution contains eight independently compilable example documents. The general overview is `kanban-board-example.tex`. The remaining files isolate deadline states, date formats, custom palettes, visual modes, column separation, page breaking, and landscape composition. This section reproduces the essential source for each case and, where visual behaviour is central, places the generated output directly below it. The resulting comparison makes the effect of each option observable without requiring the reader to compile a separate file.

9.1 Reusable document setup

A document containing several related boards should establish common values at the highest appropriate configuration level. Package options provide the initial policy, while environment options record deliberate exceptions. The following example keeps the column list, palette, mode, project start date, reference date, and column separation consistent across the document, but gives the second board a local column list and a different mode:

```
\documentclass{article}
\usepackage[
  columns={Planned,Active,Complete},
  palette=muted,
  mode=light,
  startdate=07/01/2026,
  reference-date=2026-07-18,
  columnsep=2mm
]{kanban-board}

\begin{document}

\section{Research tasks}
\begin{kanban}
  \kanbantask[prio=high,author=Alice,due=M1]
    {Planned}{Validate the data set}
  \kanbantask[author=Bob,due=M2]
    {Active}{Draft the results section}
  \kanbantask[completed]
    {Complete}{Approve the analysis plan}
\end{kanban}

\section{Archival tasks}
\begin{kanban}[columns={Queued,Processing,Stored},mode=dark]
  \kanbantask{Queued}{Deposit the source data}
  \kanbantask[author=Carol]{Processing}{Create metadata records}
\end{kanban}

\end{document}
```

The first board inherits every package default, including the configured column list. The second board inherits the `muted` palette, project start date, and column separation, while replacing its columns and visual mode for that board only. This pattern makes document-wide conventions explicit and keeps local deviations easy to identify during review.

9.2 Selecting a date format

A shared date convention should normally be established at package or setup level. A board-local selection is useful when a report reproduces material prepared for a different audience. The following source uses ISO input for every task because that form is unambiguous, then requests each available output convention independently:

```
\begin{kanban}[columns={Upcoming},date-format=us]
  \kanbantask[due=2099-07-18]
    {Upcoming}{Prepare the scheduled release}
\end{kanban}

\begin{kanban}[columns={Upcoming},date-format=english]
  \kanbantask[due=2099-07-18]
    {Upcoming}{Prepare the scheduled release}
\end{kanban}

% The same structure also accepts german, iso, and ymd.
```

Only the printed date changes. Validation and overdue classification use the same Gregorian components in all five cases.

Rendered output. The boards below render the same future date. The labels state the component order explicitly so that the result does not depend on national terminology.

US: MM/DD/YYYY

Upcoming 1

Prepare the scheduled release
07/18/2099

English: DD/MM/YYYY

Upcoming 1

Prepare the scheduled release
18/07/2099

German: DD.MM.YYYY

Upcoming 1

Prepare the scheduled release
18.07.2099

ISO: YYYY-MM-DD

Upcoming 1

Prepare the scheduled release
2099-07-18

Year first: YYYY/MM/DD

Upcoming 1

Prepare the scheduled release
2099/07/18

The complete compilable source is `kanban-board-date-formats-example.tex`. It also shows how the five compact boards can be arranged in minipages without changing their local configuration scope.

9.3 Light and dark boards in one document

Visual mode is a property of the cards rather than of the page. Selecting `dark` therefore changes card backgrounds, text, and borders without changing the surrounding document or the transparent column body. The source below keeps the column structure constant so that the change in visual treatment can be assessed directly. Author chips and neutral deadline chips keep the same light treatment in both boards. Red and amber deadline states retain their semantic colors. Two cards in each board carry a background tint, which is mixed into the surface of the respective mode.

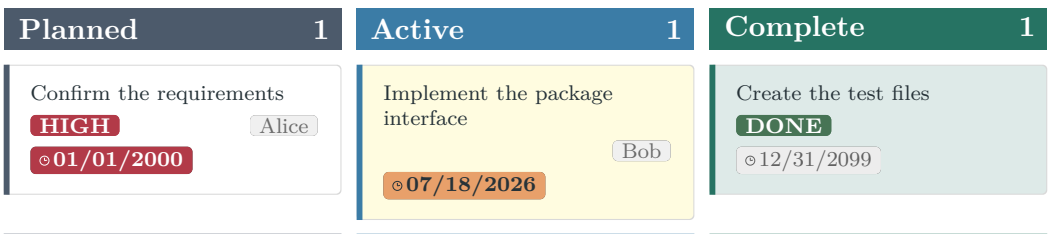
```
\begin{kanban}[columns={Planned,Active,Complete},mode=light]
  \kanbantask[prio=high,author=Alice,deadline=01/01/2000]
    {Planned}{Confirm the requirements}
  \kanbantask[bg=yellow,author=Bob,
    due=07/18/2026]
    {Active}{Implement the package interface}
  \kanbantask[bg=kanbanTeal,completed,deadline=12/31/2099]
    {Complete}{Create the test files}
\end{kanban}

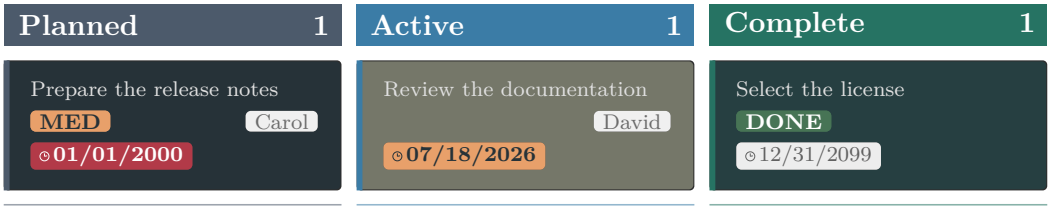
\bigskip

\begin{kanban}[columns={Planned,Active,Complete},mode=dark]
  \kanbantask[prio=medium,author=Carol,deadline=01/01/2000]
    {Planned}{Prepare the release notes}
  \kanbantask[bg=yellow,author=David,
    due=07/18/2026]
    {Active}{Review the documentation}
  \kanbantask[bg=kanbanTeal,completed,deadline=12/31/2099]
    {Complete}{Select the license}
\end{kanban}
```

A document-wide mode may instead be selected as a package option or with `\kbset`. The environment option remains the appropriate form when the change applies to one board only.

Rendered output. The following pair is generated from the preceding source. In particular, the dark board retains the selected column palette while using transparent column bodies, dark cards, light foreground text, and light neutral author and future-deadline chips matching those in the light board. Red and amber deadline chips continue to communicate status independently of visual mode.





The complete compilable source is provided as `kanban-board-modes-example.tex`.

9.4 Changing the gap between columns

Column separation follows the same three-level configuration hierarchy as the palette and mode. A package option establishes the initial value, `\kbset` changes the value for subsequent boards, and an environment option applies to one board. The following fragment demonstrates the scope of each form:

```
\usepackage[columnsep=3mm]{kanban-board}

% Uses the 3 mm package default.
\begin{kanban}[columns={Todo,Doing,Done}]
  \kanbantask[deadline=01/01/2000]{Todo}{Define the interface}
  \kanbantask[due=07/18/2026]
    {Doing}{Write the implementation}
  \kanbantask[deadline=12/31/2099,completed]{Done}{Run the tests}
\end{kanban}

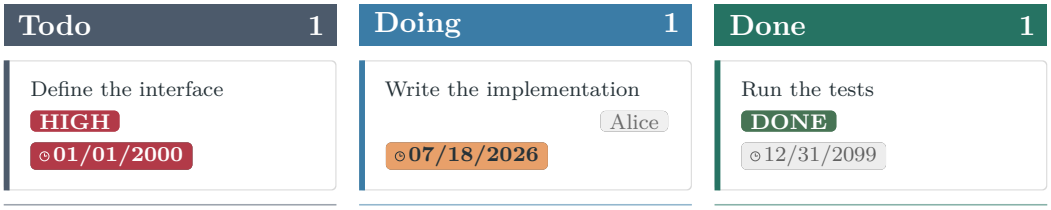
% Changes the default for subsequent boards.
\kbset{columnsep=1mm}
\begin{kanban}[columns={Todo,Doing,Done}]
  \kanbantask[deadline=01/01/2000]{Todo}{Define the interface}
  \kanbantask[due=07/18/2026]
    {Doing}{Write the implementation}
  \kanbantask[deadline=12/31/2099,completed]{Done}{Run the tests}
\end{kanban}

% Uses 4 mm here. The document default remains 1 mm.
\begin{kanban}[columns={Todo,Doing,Done},columnsep=4mm]
  \kanbantask[deadline=01/01/2000]{Todo}{Define the interface}
  \kanbantask[due=07/18/2026]
    {Doing}{Write the implementation}
  \kanbantask[deadline=12/31/2099,completed]{Done}{Run the tests}
\end{kanban}
```

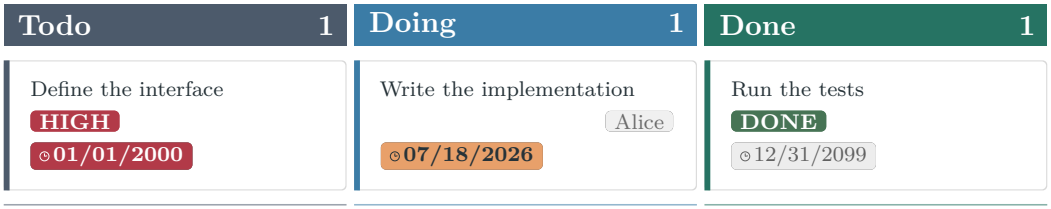
The value must be a non-negative explicit dimension such as `2mm`, a LaTeX length register such as `\linewidth`, or a TeX length parameter such as `\parindent`. Increasing it strengthens the visual separation between workflow stages, but reduces the width available to every task card.

Rendered output. In the comparison below, palette, headings, and task content are held constant. Only `columnsep` changes. Board-local values are used here so that the manual's configuration remains unchanged after the demonstration.

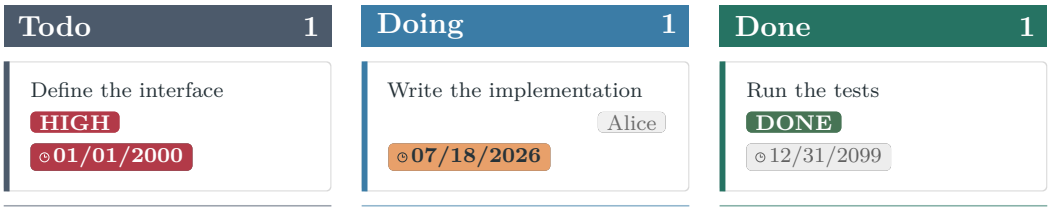
Column separation: 3 mm.



Column separation: 1 mm.



Column separation: 4 mm.



The complete source is `kanban-board-columnsep-example.tex`.

9.5 A page-breakable board

Long task lists should use the `break` option. In this layout each Kanban column is managed as a paracol column and may continue onto a later page. The fixed layout, by contrast, treats the complete board as one unbreakable object. The compact fragment below shows the public interface. The rendered demonstration that follows deliberately contains more cards so that a real page transition occurs. The synchronized headings appear once at the beginning of the board. Continuation pages contain only the remaining cards. This behavior follows the public paracol column-stream model and avoids dependencies on its private output routine.

```
\kbset{palette=default,mode=light,columnsep=2mm}
\begin{kanban}[
  columns={Backlog,In progress,Review,Done},break
]
  \kanbantask[prio=high,author=Alice,deadline=01/01/2000]
    {Backlog}{Finalize the interface}
  \kanbantask[prio=medium,author=Bob,
    due=07/18/2026]
    {In progress}{Write regression tests}
  \kanbantask[bg=yellow,author=Carol,deadline=12/31/2099]
    {Review}{Proofread the manual}
  \kanbantask[completed,due=12/31/2099]
    {Done}{Select the package license}
\end{kanban}
```

Task declarations need not follow the order of the columns. The package collects each task under its named column and preserves source order within that column. Since column bodies paginate independently, their bottom edges are not synchronized in this layout.

Rendered output. The next board begins below this paragraph and contains enough material to continue onto the following page. Its headings are printed once at the beginning. The cards that do not fit on this page continue in their respective column streams on the next page. Twelve cards carry a background tint in one of four colors, three cards per color, so the demonstration also shows that a tint travels with its card onto the continuation page.

Backlog 12	In progress 12	Review 12
Confirm the public interface. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Alice **01/01/2000**	Implement configuration keys. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Alice **07/18/2026**	Inspect the generated style file. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Alice 12/31/2099
Collect compatibility requirements. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Bob	Apply the visual modes. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Bob	Compare regression logs. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Bob
Review package dependencies. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Carol	Synchronize header heights. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Carol	Proofread the public interface. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Carol
Define the project date policy. Document the assumptions, record the outcome, and retain the evidence required for the next project review. David	Equalize fixed column bodies. Document the assumptions, record the outcome, and retain the evidence required for the next project review. David	Examine color contrast. Document the assumptions, record the outcome, and retain the evidence required for the next project review. David
Select the release palette. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Alice	Remove outer side rules. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Alice	Validate the documented source. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Alice

Identify supported engines. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Bob

Plan the documentation structure. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Carol

Prepare the release checklist. Document the assumptions, record the outcome, and retain the evidence required for the next project review. David

Confirm the archive structure. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Alice

Review installation instructions. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Bob

Assess documented limitations. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Carol

Schedule the release review. Document the assumptions, record the outcome, and retain the evidence required for the next project review. David

Extend the user manual. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Bob

Compile example documents. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Carol

Add regression assertions. Document the assumptions, record the outcome, and retain the evidence required for the next project review. David

Run the pdfTeX checks. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Alice

Run the LuaTeX checks. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Bob

Generate the release archive. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Carol

Inspect the rendered output. Document the assumptions, record the outcome, and retain the evidence required for the next project review. David

Review the CTAN metadata. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Bob

Check the license statements. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Carol

Inspect the manual PDF. Document the assumptions, record the outcome, and retain the evidence required for the next project review. David

Test the installation procedure. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Alice

Verify all example files. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Bob

Confirm the change history. Document the assumptions, record the outcome, and retain the evidence required for the next project review. Carol

Approve the release candidate. Document the assumptions, record the outcome, and retain the evidence required for the next project review. David

The continuation demonstrates why body heights remain independent when `break` is active. The complete standalone example is `kanban-board-break-example.tex`.

9.6 A fixed board on a landscape page

A landscape page increases `\linewidth` and is therefore useful when a workflow has five or more stages that would be too narrow in portrait orientation. The pdflscape package establishes the rotated page and records the appropriate PDF rotation metadata. The board itself requires no rotation commands. Page margins should be chosen before the board is opened, and `columnsep` can then be reduced modestly if the available width is still insufficient.

The board must use the fixed layout. The `break` option is deliberately absent because its paracol-based pagination is intended for ordinary portrait text pages and is not compatible with this use. A fixed board is kept as one indivisible object, including its header and body rows. It will therefore move to the next page when the remaining vertical space is too small. If the complete board is taller than one landscape page, the content should be reduced or presented as a separate page-breakable portrait board.

Concise stage names are especially important when six columns share the page. The example uses `Build` and `Docs` as familiar abbreviations for implementation and documentation. Less familiar abbreviations should be defined in the surrounding text so that the board remains understandable without relying on source-code comments.

```
\documentclass[a4paper]{article}
\usepackage[margin=18mm]{geometry}
\usepackage{pdflscape}
\usepackage[columnsep=2mm]{kanban-board}

\begin{document}
\begin{landscape}
\section*{Release workflow}

% Deliberately omit the break option in a landscape environment.
\begin{kanban}[
  columns={Ideas,Selected,Build,Review,Docs,Released},
  mode=light
]
  \kanbantask[deadline=01/01/2000]
    {Ideas}{Collect feature requests}
  \kanbantask{Ideas}{Record user feedback}
  \kanbantask[prio=high,
    due=07/18/2026]
    {Selected}{Approve the release scope}
  \kanbantask[author=Alice,deadline=12/31/2099]
    {Build}{Complete the changes}
  \kanbantask[author=Bob]{Review}{Run the acceptance review}
  \kanbantask[author=Carol]{Docs}{Update the user manual}
  \kanbantask[completed]{Released}{Publish the archive}
\end{kanban}
\end{landscape}
\end{document}
```

The supporting package is needed only by a document that requests rotated pages. It is not a dependency of kanban-board. The excerpt shows the structural pattern.

The standalone example adds three tasks to every stage so that the use of the available landscape area can be assessed realistically.

Rendered output. The next page is generated directly in a `landscape` environment. The board uses six fixed columns, short one-line headings, three cards per column, and no `break` option. Equal body heights make the workflow read as one coherent table even though individual task cards have different amounts of metadata.

Landscape release workflow

Ideas 3	Selected 3	Build 3	Review 3	Docs 3	Released 3
Collect feature requests Alice 01/01/2000	Approve the release scope HIGH Bob 07/18/2026	Complete the package changes Carol 12/31/2099	Run the technical review David	Update the user manual Alice	Publish the source archive DONE
Record user feedback Bob	Confirm supported engines Carol	Update date validation David	Compare engine output Alice	Compile usage examples Bob	Tag version 1.0.0 DONE Alice
Identify compatibility risks Carol	Assign milestone owners David	Generate the style file Alice	Check regression logs Bob	Verify CTAN metadata Carol	Announce the release DONE Bob

The output confirms that the fixed layout uses the landscape line width while retaining synchronized headings and column bodies. The task count in each heading is derived from the declarations, and the palette continues cyclically if a later board contains more stages than palette entries. A complete version is provided as `kanban-board-landscape-example.tex`.

10 License and maintenance

The package is distributed under the LaTeX Project Public License, version 1.3c or, at the user's option, any later version. Its LPPL status is "maintained", and the Current Maintainer is Andreas Willibald Weber. Defect reports should include a minimal complete document, the relevant log messages, and the versions of LaTeX and kanban-board used to reproduce the problem. The package information page is https://ctan.org/pkg/kanban-board. Development takes place at https://github.com/AndreasWillibaldWeber/kanban-board. Defects and focused enhancement proposals should be submitted through the public issue tracker at https://github.com/AndreasWillibaldWeber/kanban-board/issues. The development repository also contains the contribution guidelines and the release procedure of the maintainer.

11 Implementation

The implementation is organized as six documented parts that `kanban-board.ins` assembles into one style file. This part holds the package declaration, the dependencies, and the shared state. The remaining parts hold the palette and column registries, the deadline engine, the persistent configuration, the card renderers, and the board lifecycle. The commentary explains the responsibility of each definition and the design decisions behind it. It does not restate the semantics of the LaTeX3 programming layer, which is documented in `interface3.pdf`.

11.1 Public command contracts

The implementation distinguishes persistent defaults, one-board state, and task data. The `kanban` environment establishes a collection scope, copies the persistent defaults, applies its local options, and selects one of the two renderers. It is the only public operation that creates board state. `\kanbantask` may mutate that state only while a board is active. Otherwise it reports an error and discards the declaration. A nested board is rejected at its entry point, and its task declarations enter a temporary discard state so that the structural error is not obscured by secondary diagnostics.

`\kanbansetup` changes persistent defaults for boards that begin later. When a short name is unused, the alias is defined by control-sequence equivalence rather than by a forwarding wrapper. This applies to `\kbset`, `\kbdefinepalette`, and `\kbtask`. Existing short definitions are preserved and diagnosed. Palette definition changes

the registry. Palette selection changes the current palette name, its background list, and any paired foreground list. Redefining the selected name refreshes the persistent lists for later boards, whereas a board already in progress keeps its local snapshot. No generic `\task` command is defined, because that name has a material risk of collision with classes and unrelated packages.

11.2 Configuration and data collection

A board begins by copying the current document defaults for columns, palette, visual mode, date format, project start date, reference date, semantic labels, and column separation into board-local state. Its optional key list is then applied to that state. Every column name is trimmed. Its lookup key is the uppercase form of its plain text, with formatting commands and braces removed, and its original spelling is the heading text. A board-local property list maps each key to a consecutive numeric identifier. Task lists and counters use that identifier together with a monotonically increasing board identifier, so user-provided names never form part of an internal control-sequence name. Duplicate keys are diagnosed before rendering. Configuration is parsed by the key families `kanban/task`, `kanban/board`, and `kanban/setup`, with subordinate label families. Start and reference dates are deferred until the complete surrounding key list has resolved its date format. They are then validated and stored in canonical ISO form. Package options are accumulated and passed through the same setup transaction rather than through a second parser. Their diagnostics name the package options as the context, because the line number at that moment lies inside the package file.

Task declarations do not typeset cards immediately. Each declaration normalizes and validates the target column, increments its counter, and appends a protected call to the card renderer to the list of that column. Empty and unknown targets produce explicit errors and are not appended. Task text and metadata remain unexpanded while they are stored. Collection is global for the lifetime of the active board, so a declaration inside an ordinary TeX group is retained. The deadline of an accepted task is checked at this point, which gives every deadline warning the line number of the task. After the columns are resolved, the package evaluates the complete gap width. Dimension syntax is checked before assignment, and a value that would leave no positive width for cards is diagnosed once and replaced locally by `0pt`.

11.3 Rendering strategies

The fixed renderer creates two raster rows, one containing every heading and one containing every column body. Row-wise equalization gives all headings the height of the tallest heading and all bodies the height of the fullest body. The complete raster is enclosed in a minipage, which makes the header and body rows indivisible and prevents an orphaned header row at the foot of a page. The column body hides its frame and interior rather than painting a transparent background, so the PDF contains no transparency group and remains suitable for PDF/A workflows.

When `break` is selected, the package constructs parallel paracol columns. Header

heights are synchronized through a board-specific equal-height group, while the card lists remain outside any fixed-height box, so each column may continue on a later page. Headers are emitted once before the streams diverge and are not repeated on continuation pages. The two renderers share the same card command and visual styles but use different body-height and pagination rules.

11.4 Metadata and validation

Calendar and project-month classification is performed when a card is rendered, against the reference date of the board or the compilation date. All five supported calendar notations are reduced to the same year, month, and day components, which are checked against month lengths and the leap-year rule and then normalized for display. Every notation requires a four-digit year. Input with the shape of a date but a shorter year is reported rather than misread. Ambiguous year-last slash values follow the date-format snapshot of the board, with a validity-based swap when only the opposite ordering forms a real date. Project-month numbers are stripped of leading zeros and bounded before integer conversion, and project-month boundaries use the start-day number where possible and the last valid day in shorter months. Every other value remains ordinary label text. An optional card background is validated against the mixing grammar of xcolor and the defined color names when the task is declared, and the renderer mixes it into the surface of the active mode.

Card surfaces, metadata chips, and deadline indicators are named tcolorbox styles. Seven empty public styles are applied after the structural defaults, which permits controlled presentation changes without exposing internal environment names. Built-in header and semantic color pairs were selected against a 4.5:1 contrast threshold. Internal LaTeX2e control sequences use three prefixes. The `kanban@` prefix marks package state and helpers, the `kb@` prefix marks short-lived values of card rendering and choice normalization, and the `kbtask@` prefix marks the metadata of the task being declared. LaTeX3 variables and functions use the `__kanban` module name. The package allocates its own LaTeX3 variables for every role and does not rely on the scratch variables of the kernel. The regression suite exercises configuration scope, diagnostics, canonical date snapshots, fixed reference dates, metadata flags, malformed dimensions, compatibility with the declared LaTeX format, all three supported engines, all five date formats, custom heading foregrounds, full-width cards, synchronized break-layout headings, cross-page continuation, impossible-gap recovery, robust commands and paragraphs in user text, brace groups in column names, TeX length parameters as gaps, and a strict run with expl3 declaration and deprecation checking enabled.

11.5 Bootstrap state

The bootstrap code identifies the package, loads the packages on which rendering depends, and allocates the persistent and board-local state used by the configuration, collection, validation, and rendering stages.

Package declaration and runtime dependencies. The docstrip guard `%<*package>` begins the part of the documented source that is copied into `kanban-board.sty`. Each source part opens and closes its own guarded region, and `kanban-board.ins` assembles the parts in dependency order. The package declares the minimum LaTeX format and loads each facility that its public behavior requires rather than relying on another package to load that facility incidentally. xcolor provides named colors and color expressions. etoolbox supplies Booleans, string tests, and expandable list construction. tikz draws the clock symbol used by deadlines. tcolorbox constructs headers, cards, chips, and the fixed-board raster, with the `skins` and `raster` libraries. paracol provides the independent page-breakable column streams of the `break` renderer.

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}[2020-10-01]
\ProvidesPackage{kanban-board}[2026/11/15 v1.0.0 Kanban project boards]
\RequirePackage{xcolor}
\RequirePackage{etoolbox}
\RequirePackage{tikz}
\RequirePackage{tcolorbox}
\RequirePackage{paracol}
\tcbuselibrary{skins,raster}
```

Built-in color constants. The semantic colors and the colors of the built-in palettes are defined once under stable symbolic names. The three semantic colors represent overdue or high-priority work, work due at the reference date, and completed or low-priority work. The other names are palette entries. Text colors are not hard-coded into the palette registry here, because a starred palette definition may provide an explicit foreground for each background.

```
\definecolor{kanbanInk}{HTML}{263238}
\definecolor{kanbanHigh}{HTML}{B23A48}
\definecolor{kanbanMed}{HTML}{E8A06A}
\definecolor{kanbanLow}{HTML}{477455}
\definecolor{kanbanSlate}{HTML}{4C5A6B}
\definecolor{kanbanBlue}{HTML}{3B7CA6}
\definecolor{kanbanTeal}{HTML}{267363}
\definecolor{kanbanSage}{HTML}{4F7D43}
\definecolor{kanbanAmber}{HTML}{876515}
\definecolor{kanbanClay}{HTML}{9B5030}
\definecolor{kanbanRose}{HTML}{AF5C74}
\definecolor{kanbanPlum}{HTML}{7C5C95}
\definecolor{kanbanMutedBlue}{HTML}{52758C}
\definecolor{kanbanMutedGreen}{HTML}{597C6E}
\definecolor{kanbanMutedPurple}{HTML}{746680}
\definecolor{kanbanMutedBrown}{HTML}{8A6F5A}
```

Boolean flags, board identity, and dimensions. Rendering needs a small set of state flags. They record whether a board uses the page-breakable renderer, whether task declarations currently belong to an active board, whether the environment should render when it closes, and whether tasks inside a rejected nested board must be discarded. Two further flags carry the result of deadline classification. A unique

board number keeps the temporary data of different boards apart, and dedicated length registers hold the persistent gap, its board-local snapshot, a validation candidate, and the combined width of all gaps. Separate registers let validation inspect a candidate without modifying a previously accepted value.

```
\newbool{kanban@break}
\newbool{kanban@active}
\newbool{kanban@render}
\newbool{kanban@discardtasks}
\newbool{kb@overdue}
\newbool{kb@duenow}
\newcounter{kanban@board}
\def\kanban@boardgroup{0}
\newlength\kanban@columnsep
\newlength\kanban@boardcolumnsep
\newlength\kanban@candidatecolumnsep
\newlength\kanban@totalgaps
\setlength\kanban@columnsep{2mm}
```

Persistent defaults and board-local scalar snapshots. Configuration values such as the start date, reference date, date format, visual mode, card colors, and semantic labels are simple token values. Most occur in pairs. One member is the persistent default for later boards, and the other is the snapshot used by the board currently being rendered. This separation implements the documented package, setup, and board precedence. Empty date macros mean that no fixed date has been supplied, in which case the compilation date is the reference. The color macros contain a named color or an xcolor expression and are expanded only when a box is rendered.

```
\def\kanban@startdate{}
\def\kanban@thestart{}
\def\kanban@referencedate{}
\def\kanban@thereference{}
\def\kanban@dateformat{us}
\def\kanban@boarddateformat{us}
\def\kanban@mode{light}
\def\kanban@boardmode{light}
\def\kanban@textcolor{kanbanInk}
\def\kanban@cardcolor{white}
\def\kanban@cardtint{15}
\def\kanban@bordercolor{black!15}
\def\kanban@authorback{black!6}
\def\kanban@authortext{black!58}
\def\kanban@deadlineback{black!7}
\def\kanban@deadlinetext{black!62}
\def\kanban@headingtext{white}
\def\kanban@hightext{white}
\def\kanban@mediumtext{kanbanInk}
\def\kanban@lowtext{white}
\def\kanban@highlabel{HIGH}
\def\kanban@mediumlabel{MED}
\def\kanban@lowlabel{LOW}
\def\kanban@donelabel{DONE}
\def\kanban@boardhighlabel{HIGH}
\def\kanban@boardmediumlabel{MED}
\def\kanban@boardlowlabel{LOW}
```

```
\def\kanban@boarddonelabel{DONE}
```

LaTeX3 state containers. Palettes and columns require structured data rather than isolated scalar macros, and every role receives its own typed variable. The two global property lists form the palette registry, one for heading backgrounds and one for optional heading foregrounds. Global comma lists hold the selected persistent palette and the default column keys, and a global property list maps each column key to its heading text. Their local counterparts are the snapshot for an active board. The remaining local containers support palette-pair validation, column normalization, numeric column lookup, normalized choices, and the deferred processing of start and reference dates. The scratch variables of the kernel are not used, because other packages may overwrite them between calls.

```
\ExplSyntaxOn
\prop_new:N \g__kanban_palettes_prop
\prop_new:N \g__kanban_palette_texts_prop
\clist_new:N \g__kanban_palette_clist
\clist_new:N \g__kanban_palette_fg_clist
\tl_new:N \g__kanban_palette_name_tl
\clist_new:N \l__kanban_board_palette_clist
\clist_new:N \l__kanban_board_palette_fg_clist
\clist_new:N \l__kanban_palette_colors_clist
\clist_new:N \l__kanban_palette_bg_clist
\clist_new:N \l__kanban_palette_fg_clist
\tl_new:N \l__kanban_palette_bg_tl
\tl_new:N \l__kanban_palette_fg_tl
\tl_new:N \l__kanban_pair_bg_tl
\tl_new:N \l__kanban_pair_fg_tl
\tl_new:N \l__kanban_palette_item_tl
\int_new:N \l__kanban_palette_count_int
\int_new:N \l__kanban_palette_index_int
\seq_new:N \l__kanban_palette_pair_seq
\bool_new:N \l__kanban_bad_palette_bool
\clist_new:N \g__kanban_columns_clist
\prop_new:N \g__kanban_column_display_prop
\clist_new:N \l__kanban_board_columns_clist
\prop_new:N \l__kanban_board_column_display_prop
\clist_new:N \l__kanban_columns_tmp_clist
\prop_new:N \l__kanban_column_display_tmp_prop
\prop_new:N \l__kanban_column_ids_prop
\int_new:N \l__kanban_column_index_int
\tl_new:N \l__kanban_palette_name_tl
\tl_new:N \l__kanban_column_tl
\tl_new:N \l__kanban_column_display_tl
\tl_new:N \l__kanban_column_id_tl
\tl_new:N \l__kanban_choice_tl
\tl_new:N \l__kanban_background_tl
\seq_new:N \l__kanban_background_seq
\bool_new:N \l__kanban_background_ok_bool
\tl_new:N \l__kanban_pending_startdate_tl
\tl_new:N \l__kanban_pending_reference_date_tl
\bool_new:N \l__kanban_pending_startdate_bool
\bool_new:N \l__kanban_pending_reference_date_bool
```

Named diagnostics. Every warning of the package is a named message, so the main text and the help text stay consistent whether invalid input originates in a package option, in `\kbset`, in a board option, in a palette definition, or in a task declaration. Each message states the line context after the subject of its first sentence, because l3msg warnings do not add it on their own. Messages that share a shape take the scope or the kind of the setting as a parameter. The alias message is issued while the package loads and therefore carries no line context. A message text that continues on a second source line ends the first line with a tilde. The tilde is a space character in expl3 syntax, and TeX discards a space character at the start of a line. For the same reason a tilde directly after `\msg_line_context:` is absorbed as the end of that control word, so a word space at that position is written as `\c_space_tl`. The LaTeX2e-facing wrapper lets the parts written in LaTeX2e syntax issue a named message with two parameters. A call site that needs an expanded argument, such as the name of an unknown key, expands the complete call with `\use:x` and then invokes the base message function. The x-type message variants are separate definitions on older kernels, so a call through them would bypass a test harness that intercepts the base function.

```
\msg_new:nnnn {kanban-board}{invalid-date-setting}
  {Invalid~#1~'#2'~\msg_line_context:.~The~current~value~is~kept.}
  {Use~a~valid~Gregorian~date~with~a~four-digit~year~in~one~of~the~
    documented~numeric~forms,~for~example~2026-07-18.}
\msg_new:nnnn {kanban-board}{invalid-dimension}
  {Invalid~#1~value~'#2'~\msg_line_context:.~The~current~value~
    is~kept.}
  {Use~a~non-negative~explicit~dimension~such~as~2mm,~a~LaTeX~length~
    register,~or~a~TeX~length~parameter~such~as~\iow_char:N\\hsize.}
\msg_new:nnnn {kanban-board}{unknown-key}
  {Unknown~#1~option~'#2'~\msg_line_context:.~The~option~is~ignored.}
  {Check~the~spelling~and~the~documented~scope~of~the~key.}
\msg_new:nnnn {kanban-board}{unknown-value}
  {Unknown~#1~'#2'~\msg_line_context:.~The~current~value~is~kept.}
  {Check~the~spelling~of~the~value~against~the~documented~choices.}
\msg_new:nnnn {kanban-board}{unknown-palette}
  {Unknown~#1~'#2'~\msg_line_context:.~The~current~palette~is~kept.}
  {Define~the~palette~with~\iow_char:N\\kanbandefinepalette~before~
    selecting~it.~The~built-in~names~are~default,~muted,~and~
    monochrome.}
\msg_new:nnnn {kanban-board}{empty-palette-name}
  {Empty~palette~name~\msg_line_context:.~The~definition~is~ignored.}
  {Give~the~palette~a~non-empty~name.}
\msg_new:nnnn {kanban-board}{empty-palette}
  {Palette~'#1'~\msg_line_context:\c_space_tl has~no~#2.~
    The~definition~is~ignored.}
  {List~at~least~one~entry,~for~example~kanbanBlue~or~
    kanbanBlue/white.}
\msg_new:nnnn {kanban-board}{malformed-palette-pair}
  {Palette~'#1'~\msg_line_context:\c_space_tl has~a~malformed~
    color/text~pair.~The~definition~is~ignored.}
  {Write~every~entry~as~background/foreground~with~two~non-empty~
    color~names,~for~example~kanbanSlate/white.}
\msg_new:nnnn {kanban-board}{empty-column-list}
  {Empty~column~list~\msg_line_context:.~The~current~list~is~kept.}
  {Give~at~least~one~column~name,~for~example~columns={Backlog,Done}.}
\msg_new:nnnn {kanban-board}{empty-label}
  {Empty~#1~'#2'~\msg_line_context:.~The~current~value~is~kept.}
  {Give~every~label~a~non-empty~text,~for~example~
```

```
    labels={high=URGENT}.}
\msg_new:nnnn {kanban-board}{unknown-color}
  {Unknown~color~'#1'~\msg_line_context:.~The~card~keeps~the~default~
    background.}
  {Use~a~color~name~known~to~xcolor,~or~an~expression~such~as~
    yellow!50~or~red!30!blue~whose~names~are~all~known.}
\msg_new:nnnn {kanban-board}{unknown-priority}
  {Unknown~priority~'#1'~\msg_line_context:.~The~priority~chip~is~
    omitted.}
  {Use~high,~medium,~med,~or~low.}
\msg_new:nnnn {kanban-board}{invalid-deadline}
  {Invalid~deadline~date~'#1'~\msg_line_context:.~The~card~shows~a~
    neutral~label.}
  {Write~calendar~deadlines~with~a~four-digit~year~in~one~of~the~
    documented~numeric~forms,~for~example~2026-07-18.}
\msg_new:nnnn {kanban-board}{invalid-project-month}
  {Invalid~project-month~deadline~'#1'~\msg_line_context:.~The~card~
    shows~a~neutral~label.}
  {Write~project~months~as~M~followed~by~an~integer~from~1~to~999999.}
\msg_new:nnnn {kanban-board}{alias-taken}
  {\iow_char:N\\#1~is~already~defined.~Use~\iow_char:N\\#2~instead.}
  {Another~package~or~the~document~owns~the~short~name.~The~existing~
    definition~is~kept.}
\cs_new_protected:Npn \kanban@warning #1#2#3
  {\msg_warning:nnnn {kanban-board}{#1}{#2}{#3}}
```

At this point the package has allocated, but not yet populated, the palette registries and the column maps. The following parts define the functions that validate and populate those containers.

```
⟨/package⟩
```

```
⟨*package⟩
```

11.6 Palette and column registries

Palette names map to lists of heading backgrounds and, for starred definitions, to paired lists of heading foregrounds. Both maps are global property lists. Column names are normalized separately and receive board-local numeric identifiers, so user text never becomes part of an internal control-sequence name. The registry therefore has two responsibilities. It maintains the persistent palette catalogue, and it converts the textual column interface into the compact board-local representation that the renderers consume.

Plain palette definitions. An unstarred definition accepts a name and a non-empty comma list of heading backgrounds. A successful definition replaces any palette of the same name and removes a previously stored foreground list. If the redefined name is the selected palette, the persistent snapshot is refreshed immediately so that later boards use the new colors without a second selection. An invalid definition leaves every registry unchanged.

```
\cs_new_protected:Npn \__kanban_palette_define:nn #1#2
  {
    \tl_set:Nn \l__kanban_palette_name_tl {#1}
    \tl_trim_spaces:N \l__kanban_palette_name_tl
    \tl_if_blank:VTF \l__kanban_palette_name_tl
      {\msg_warning:nn {kanban-board}{empty-palette-name}}
      {
        \clist_set:Nn \l__kanban_palette_colors_clist {#2}
        \clist_if_empty:NTF \l__kanban_palette_colors_clist
          {\use:x {\msg_warning:nnnn {kanban-board}{empty-palette}
            {\exp_not:V \l__kanban_palette_name_tl}{colors}}}
          {
            \prop_gput:NVV \g__kanban_palettes_prop
              \l__kanban_palette_name_tl \l__kanban_palette_colors_clist
            \prop_gremove:NV \g__kanban_palette_texts_prop
              \l__kanban_palette_name_tl
            \tl_if_eq:NNT \l__kanban_palette_name_tl
              \g__kanban_palette_name_tl
              {
                \clist_gset_eq:NN \g__kanban_palette_clist
                  \l__kanban_palette_colors_clist
                \clist_gclear:N \g__kanban_palette_fg_clist
              }
          }
      }
  }
```

Paired palette definitions. The starred syntax associates every heading background with an explicit heading foreground. Each comma item must contain exactly one / separator and two non-empty components. The item helper validates one pair and appends its components to the working lists, or records the failure. The definition is committed only after every pair has passed. A single malformed item rejects the whole palette rather than installing a partial result.

```
\cs_new_protected:Npn \__kanban_palette_pair:n #1
  {
    \seq_set_split:Nnn \l__kanban_palette_pair_seq {/} {#1}
    \int_compare:nNnTF {\seq_count:N \l__kanban_palette_pair_seq} = {2}
      {
        \tl_set:Nx \l__kanban_pair_bg_tl
          {\seq_item:Nn \l__kanban_palette_pair_seq {1}}
        \tl_set:Nx \l__kanban_pair_fg_tl
          {\seq_item:Nn \l__kanban_palette_pair_seq {2}}
        \tl_trim_spaces:N \l__kanban_pair_bg_tl
        \tl_trim_spaces:N \l__kanban_pair_fg_tl
        \bool_if:nTF
          {
            \tl_if_blank_p:V \l__kanban_pair_bg_tl
            || \tl_if_blank_p:V \l__kanban_pair_fg_tl
          }
          {\bool_set_true:N \l__kanban_bad_palette_bool}
          {
            \clist_put_right:NV \l__kanban_palette_bg_clist
              \l__kanban_pair_bg_tl
            \clist_put_right:NV \l__kanban_palette_fg_clist
              \l__kanban_pair_fg_tl
          }
      }
```

```
      {\bool_set_true:N \l__kanban_bad_palette_bool}
  }
\cs_new_protected:Npn \__kanban_palette_define_pairs:nn #1#2
  {
    \tl_set:Nn \l__kanban_palette_name_tl {#1}
    \tl_trim_spaces:N \l__kanban_palette_name_tl
    \clist_clear:N \l__kanban_palette_bg_clist
    \clist_clear:N \l__kanban_palette_fg_clist
    \bool_set_false:N \l__kanban_bad_palette_bool
    \tl_if_blank:VTF \l__kanban_palette_name_tl
      {\msg_warning:nn {kanban-board}{empty-palette-name}}
      {
        \clist_set:Nn \l__kanban_palette_colors_clist {#2}
        \clist_if_empty:NTF \l__kanban_palette_colors_clist
          {\use:x {\msg_warning:nnnn {kanban-board}{empty-palette}
            {\exp_not:V \l__kanban_palette_name_tl}{color/text~pairs}}}
          {
            \clist_map_function:NN \l__kanban_palette_colors_clist
              \__kanban_palette_pair:n
            \bool_if:NTF \l__kanban_bad_palette_bool
              {\use:x {\msg_warning:nnn {kanban-board}{malformed-palette-pair}
                {\exp_not:V \l__kanban_palette_name_tl}}}
              {
                \prop_gput:NVV \g__kanban_palettes_prop
                  \l__kanban_palette_name_tl
                  \l__kanban_palette_bg_clist
                \prop_gput:NVV \g__kanban_palette_texts_prop
                  \l__kanban_palette_name_tl
                  \l__kanban_palette_fg_clist
                \tl_if_eq:NNT \l__kanban_palette_name_tl
                  \g__kanban_palette_name_tl
                  {
                    \clist_gset_eq:NN \g__kanban_palette_clist
                      \l__kanban_palette_bg_clist
                    \clist_gset_eq:NN \g__kanban_palette_fg_clist
                      \l__kanban_palette_fg_clist
                  }
              }
          }
      }
  }
```

Persistent selection and board-local selection. Selecting a palette is distinct from defining one. Document-wide setup replaces the persistent defaults used by future boards. A board option replaces only the snapshot of the current board. Both selectors verify the requested name against the same global registry and keep the current selection when the name is unknown. The board initializer takes the initial local snapshot when a board begins.

```
\cs_new_protected:Npn \__kanban_palette_select_global:n #1
  {
    \tl_set:Nn \l__kanban_palette_name_tl {#1}
    \tl_trim_spaces:N \l__kanban_palette_name_tl
    \prop_get:NVNTF \g__kanban_palettes_prop
      \l__kanban_palette_name_tl \l__kanban_palette_bg_tl
      {
        \clist_gset:NV \g__kanban_palette_clist \l__kanban_palette_bg_tl
        \prop_get:NVNTF \g__kanban_palette_texts_prop
```

```
          \l__kanban_palette_name_tl \l__kanban_palette_fg_tl
          {\clist_gset:NV \g__kanban_palette_fg_clist
            \l__kanban_palette_fg_tl}
          {\clist_gclear:N \g__kanban_palette_fg_clist}
        \tl_gset_eq:NN \g__kanban_palette_name_tl
          \l__kanban_palette_name_tl
      }
      {\use:x {\msg_warning:nnnn {kanban-board}{unknown-palette}
        {palette}{\exp_not:V \l__kanban_palette_name_tl}}}
  }
\cs_new_protected:Npn \__kanban_palette_select_board:n #1
  {
    \tl_set:Nn \l__kanban_palette_name_tl {#1}
    \tl_trim_spaces:N \l__kanban_palette_name_tl
    \prop_get:NVNTF \g__kanban_palettes_prop
      \l__kanban_palette_name_tl \l__kanban_palette_bg_tl
      {
        \clist_set:NV \l__kanban_board_palette_clist
          \l__kanban_palette_bg_tl
        \prop_get:NVNTF \g__kanban_palette_texts_prop
          \l__kanban_palette_name_tl \l__kanban_palette_fg_tl
          {\clist_set:NV \l__kanban_board_palette_fg_clist
            \l__kanban_palette_fg_tl}
          {\clist_clear:N \l__kanban_board_palette_fg_clist}
      }
      {\use:x {\msg_warning:nnnn {kanban-board}{unknown-palette}
        {board~palette}{\exp_not:V \l__kanban_palette_name_tl}}}
  }
\cs_new_protected:Npn \kanban@paletteboardinit
  {
    \clist_set_eq:NN \l__kanban_board_palette_clist
      \g__kanban_palette_clist
    \clist_set_eq:NN \l__kanban_board_palette_fg_clist
      \g__kanban_palette_fg_clist
  }
```

Cyclic palette lookup. A board may contain more columns than its palette contains colors. The lookup therefore maps each one-based column number cyclically onto the board palette and publishes both the background and its heading foreground through two LaTeX2e-facing macros. If the selected palette has no explicit foreground list, the heading color of the board theme is used.

```
\cs_new_protected:Npn \kanban@palettecolor #1
  {
    \int_set:Nn \l__kanban_palette_count_int
      {\clist_count:N \l__kanban_board_palette_clist}
    \int_set:Nn \l__kanban_palette_index_int
      {1 + \int_mod:nn {#1 - 1} {\l__kanban_palette_count_int}}
    \tl_set:Nx \l__kanban_palette_item_tl
      {
        \clist_item:Nn \l__kanban_board_palette_clist
          {\l__kanban_palette_index_int}
      }
    \cs_set:Npx \kb@thiscolor {\tl_use:N \l__kanban_palette_item_tl}
    \clist_if_empty:NTF \l__kanban_board_palette_fg_clist
      {\cs_set:Npn \kb@thisheadingtext {\kanban@headingtext}}
      {
        \tl_set:Nx \l__kanban_palette_item_tl
```

```
          {
            \clist_item:Nn \l__kanban_board_palette_fg_clist
              {\l__kanban_palette_index_int}
          }
        \cs_set:Npx \kb@thisheadingtext
          {\tl_use:N \l__kanban_palette_item_tl}
      }
  }
```

\kanbandefinepalette
\kbdefinepalette

The public definition command exposes both registry formats through one interface. The unstarred form accepts a list of backgrounds and the starred form accepts background/foreground pairs. The long command is the stable public name. The concise alias is installed only when that name is free, so loading kanban-board never overwrites a command owned by another package.

```
\NewDocumentCommand \kanbandefinepalette {smm}
  {
    \IfBooleanTF{#1}
      {\__kanban_palette_define_pairs:nn {#2}{#3}}
      {\__kanban_palette_define:nn {#2}{#3}}
  }
\cs_if_exist:NTF \kbdefinepalette
  {\msg_warning:nnnn {kanban-board}{alias-taken}
    {kbdefinepalette}{kanbandefinepalette}}
  {\cs_new_eq:NN \kbdefinepalette \kanbandefinepalette}
```

Internal selection bridges. The setup and board-key layers are written partly in LaTeX2e syntax. These wrappers give those layers stable names for the selection functions and add no state or validation of their own.

```
\cs_new_protected:Npn \kanban@selectpalette #1
  {\__kanban_palette_select_global:n {#1}}
\cs_new_protected:Npn \kanban@selectboardpalette #1
  {\__kanban_palette_select_board:n {#1}}
```

Column-name normalization. Column identity is case-insensitive and ignores formatting, but the visible heading keeps the spelling given in the column list. The single-name function therefore produces two results. It stores the trimmed input as the display text and writes the lookup key into the destination variable. The key is the purified text of the name in uppercase. Ordinary macros are expanded, and formatting commands and braces are removed. A column declared as `\emph{Soon}` is therefore addressed as `soon` by a task, and the heading still shows the emphasized text. Purification also keeps robust commands out of the expanded definitions that later publish the key list. The list function applies that rule to every item, ignores blank items, records the first spelling of each key in a display map, and diagnoses duplicates after normalization. Membership is tested in the display map rather than in the key list, because a property list treats every key as a string, whereas the comma list test does not support items that contain a brace group. A duplicate is a user error rather than a recoverable preference, so it produces an error with corrective help, and the first occurrence is retained.

```
\cs_new_protected:Npn \__kanban_column_normalize:nN #1#2
  {
    \tl_set:Nn \l__kanban_column_display_tl {#1}
    \tl_trim_spaces:N \l__kanban_column_display_tl
    \tl_set:Nx #2
      {\exp_args:NV \text_purify:n \l__kanban_column_display_tl}
    \tl_set:Nx #2 {\exp_args:NV \text_uppercase:n #2}
    \tl_trim_spaces:N #2
  }
\cs_new_protected:Npn \__kanban_columns_normalize:nNN #1#2#3
  {
    \clist_clear:N #2
    \prop_clear:N #3
    \clist_map_inline:nn {#1}
      {
        \__kanban_column_normalize:nN {##1} \l__kanban_column_tl
        \tl_if_blank:VF \l__kanban_column_tl
          {
            \prop_if_in:NVTF #3 \l__kanban_column_tl
              {\PackageError{kanban-board}
                {duplicate~kanban~column~
                  '\tl_use:N \l__kanban_column_tl'}
                {Column~names~are~case-insensitive.\MessageBreak
                  Remove~the~repeated~name~from~the~column~list.}}
              {
                \clist_put_right:NV #2 \l__kanban_column_tl
                \prop_put:NVV #3 \l__kanban_column_tl
                  \l__kanban_column_display_tl
              }
          }
      }
  }
```

Persistent and board-local column lists. As with palettes, columns have a persistent default and a board-local snapshot. Each consists of the normalized key list and the display map. Document setup accepts only a non-empty normalized list and makes both parts global. Board initialization copies the defaults locally. A non-blank board override replaces the copy without affecting later boards.

```
\cs_new_protected:Npn \__kanban_columns_select_global:n #1
  {
    \__kanban_columns_normalize:nNN {#1}
      \l__kanban_columns_tmp_clist
      \l__kanban_column_display_tmp_prop
    \clist_if_empty:NTF \l__kanban_columns_tmp_clist
      {\msg_warning:nn {kanban-board}{empty-column-list}}
      {
        \clist_gset_eq:NN \g__kanban_columns_clist
          \l__kanban_columns_tmp_clist
        \prop_gset_eq:NN \g__kanban_column_display_prop
          \l__kanban_column_display_tmp_prop
      }
  }
\cs_new_protected:Npn \kanban@selectcolumns #1
  {\__kanban_columns_select_global:n {#1}}
\cs_new_protected:Npn \kanban@columnsboardinit
  {
    \clist_set_eq:NN \l__kanban_board_columns_clist
```

```
      \g__kanban_columns_clist
    \prop_set_eq:NN \l__kanban_board_column_display_prop
      \g__kanban_column_display_prop
  }
\cs_new_protected:Npn \kanban@selectboardcolumns #1
  {
    \tl_if_blank:nF {#1}
      {
        \__kanban_columns_normalize:nNN {#1}
          \l__kanban_columns_tmp_clist
          \l__kanban_column_display_tmp_prop
        \clist_if_empty:NF \l__kanban_columns_tmp_clist
          {
            \clist_set_eq:NN \l__kanban_board_columns_clist
              \l__kanban_columns_tmp_clist
            \prop_set_eq:NN \l__kanban_board_column_display_prop
              \l__kanban_column_display_tmp_prop
          }
      }
  }
```

Task-target resolution. A task names a column textually, but collection must append the task to a numeric board-local list. The resolver normalizes the supplied name, exposes the normalized text for diagnostics, and looks it up in the name-to-identifier map of the active board. A missing key leaves the identifier empty, which lets the collection layer issue the context-specific error.

```
\cs_new_protected:Npn \kanban@normtaskcol #1
  {
    \__kanban_column_normalize:nN {#1} \l__kanban_column_tl
    \cs_set:Npx \kbtask@column {\tl_use:N \l__kanban_column_tl}
    \cs_set:Npn \kbtask@columnid { }
    \prop_get:NVNTF
      \l__kanban_column_ids_prop \l__kanban_column_tl
      \l__kanban_column_id_tl
      {\cs_set:Npx \kbtask@columnid
        {\tl_use:N \l__kanban_column_id_tl}}
      { }
  }
```

Numeric column registration. Before tasks can be collected, every normalized board column receives a consecutive identifier. The setup function also prepares the printable list of valid keys for error messages and delegates the creation of per-column counters, lists, colors, and renderer entries to `\kanban@setupcolumn`, which is defined in the board part. The printable list is expanded through `\text_expand:n`, which expands the comma list but leaves any robust command intact, so the expanded definition cannot be derailed. The display text is passed by value so that it is stored unexpanded for the heading.

```
\cs_new_protected:Npn \kanban@setupboardcolumns
  {
    \cs_set:Npx \kanban@availablecolumns
      {
        \text_expand:n
```

```
          {\clist_use:Nn \l__kanban_board_columns_clist {,}}
      }
    \prop_clear:N \l__kanban_column_ids_prop
    \int_zero:N \l__kanban_column_index_int
    \clist_map_inline:Nn \l__kanban_board_columns_clist
      {
        \int_incr:N \l__kanban_column_index_int
        \tl_set:Nx \l__kanban_column_id_tl
          {\int_use:N \l__kanban_column_index_int}
        \prop_put:NnV \l__kanban_column_ids_prop {##1}
          \l__kanban_column_id_tl
        \prop_get:NnN \l__kanban_board_column_display_prop {##1}
          \l__kanban_column_display_tl
        \exp_args:NVV \kanban@setupcolumn
          \l__kanban_column_display_tl \l__kanban_column_id_tl
      }
  }
```

Choice normalization and registry cleanup. Several key values, including modes and date-format names, share the same case-insensitive choice syntax. The choice helper trims and lowercases such a value before publishing it as `\kb@choice`. The lowercase conversion expands ordinary macros and leaves robust commands intact, and the published copy is not expanded a second time. A robust command in a choice is therefore reported as an unknown value instead of derailing the definition. At the end of the board lifecycle, the cleanup function removes all dynamically created per-column control sequences so that completed boards do not retain card material or counters in the hash table. The names are constructed from character data, so the `c` argument type is required.

```
\cs_new_protected:Npn \kanban@normalizechoice #1
  {
    \tl_set:Nn \l__kanban_choice_tl {#1}
    \tl_trim_spaces:N \l__kanban_choice_tl
    \tl_set:Nx \l__kanban_choice_tl
      {\exp_args:NV \text_lowercase:n \l__kanban_choice_tl}
    \cs_set:Npx \kb@choice {\exp_not:V \l__kanban_choice_tl}
  }
\cs_new_protected:Npn \kanban@cleanupboard
  {
    \int_step_inline:nn {\kanban@ncols}
      {
        \cs_undefine:c {kanban@name@##1}
        \cs_undefine:c {kanban@color@##1}
        \cs_undefine:c {kanban@headingtext@##1}
        \cs_undefine:c
          {kanban@list@\kanban@boardgroup @##1}
        \cs_undefine:c
          {kanban@count@\kanban@boardgroup @##1}
      }
  }
```

```
⟨/package⟩
```

```
473 ⟨*package⟩
```

11.7 Deadline engine

Deadline classification uses anchored regular expressions before any numeric conversion. The five supported notations enter one component validator and one formatter. Bounded project-month labels use a separate validated path, and arbitrary descriptive labels remain neutral and unchanged. The engine separates syntax recognition, Gregorian validation, display formatting, and temporal classification. This separation lets a task keep a neutral descriptive deadline without misclassifying it as a broken date. Diagnostics are issued when a task is declared, so a warning names the line of the task rather than the end of the board.

Typed parsing and calendar state. Date processing needs the normalized input, the capture groups of the last match, numeric components, maximum-day calculations, project-month state, reference-date state, display fragments, and explicit status flags. Each role has its own variable, so no scratch register carries two meanings across nested tests. The scratch variables of the kernel are deliberately not used, because other packages may overwrite them between calls.

```
474 \tl_new:N \l__kanban_deadline_tl
475 \seq_new:N \l__kanban_deadline_match_seq
476 \int_new:N \l__kanban_deadline_day_int
477 \int_new:N \l__kanban_deadline_month_int
478 \int_new:N \l__kanban_deadline_year_int
479 \int_new:N \l__kanban_deadline_max_day_int
480 \int_new:N \l__kanban_swap_int
481 \int_new:N \l__kanban_deadline_key_int
482 \int_new:N \l__kanban_reference_key_int
483 \int_new:N \l__kanban_project_month_int
484 \int_new:N \l__kanban_current_month_int
485 \int_new:N \l__kanban_project_start_day_int
486 \int_new:N \l__kanban_project_start_month_int
487 \int_new:N \l__kanban_project_start_year_int
488 \int_new:N \l__kanban_project_boundary_day_int
489 \int_new:N \l__kanban_reference_day_int
490 \int_new:N \l__kanban_reference_month_int
491 \int_new:N \l__kanban_reference_year_int
492 \tl_new:N \l__kanban_reference_tl
493 \tl_new:N \l__kanban_display_day_tl
494 \tl_new:N \l__kanban_display_month_tl
495 \tl_new:N \l__kanban_display_year_tl
496 \tl_new:N \l__kanban_display_format_tl
497 \tl_new:N \l__kanban_dimension_tl
498 \bool_new:N \l__kanban_valid_date_bool
499 \bool_new:N \l__kanban_valid_project_month_bool
500 \bool_new:N \l__kanban_date_syntax_bool
501 \bool_new:N \l__kanban_date_components_bool
502 \bool_new:N \l__kanban_ambiguous_slash_bool
503 \bool_new:N \l__kanban_project_month_syntax_bool
```

The two syntax flags have distinct meanings. The syntax flag records that the input

has the shape of a date, which is the condition for a warning. The components flag records that one of the exact notations captured year, month, and day, which is the condition for validation.

Maximum days in a Gregorian month. Component validation needs the number of permitted days in the parsed month. February has 29 days in years divisible by 400, or divisible by 4 but not by 100. April, June, September, and November have 30 days. All other valid months have 31. The divisibility predicate reads the year component that the parser has already assigned.

```
\prg_new_conditional:Npnn \__kanban_year_divisible:n #1 {p}
  {
    \int_compare:nNnTF
      {\int_mod:nn {\l__kanban_deadline_year_int} {#1}} = {0}
      {\prg_return_true:}
      {\prg_return_false:}
  }
\cs_new_protected:Npn \__kanban_set_max_day:
  {
    \int_case:nnF {\l__kanban_deadline_month_int}
      {
        {2}
          {
            \bool_if:nTF
              {
                \__kanban_year_divisible_p:n {400}
                || (
                     \__kanban_year_divisible_p:n {4}
                     && ! \__kanban_year_divisible_p:n {100}
                   )
              }
              {\int_set:Nn \l__kanban_deadline_max_day_int {29}}
              {\int_set:Nn \l__kanban_deadline_max_day_int {28}}
          }
        {4}{\int_set:Nn \l__kanban_deadline_max_day_int {30}}
        {6}{\int_set:Nn \l__kanban_deadline_max_day_int {30}}
        {9}{\int_set:Nn \l__kanban_deadline_max_day_int {30}}
        {11}{\int_set:Nn \l__kanban_deadline_max_day_int {30}}
      }
      {\int_set:Nn \l__kanban_deadline_max_day_int {31}}
  }
```

Gregorian component validation and capture conversion. Once a notation has assigned year, month, and day, the validator checks that the year is positive, that the month lies between 1 and 12, and that the day lies in the range permitted by the month and the leap-year rule. The tests are nested so that the maximum-day helper never sees an invalid month. The capture helpers provide the single conversion path from regular-expression text to integers. The three-argument helper receives the capture positions of year, month, and day and also records that the input had both date syntax and complete components.

```
\cs_new_protected:Npn \__kanban_validate_date:
  {
    \bool_set_false:N \l__kanban_valid_date_bool
```

```
    \int_compare:nNnT {\l__kanban_deadline_year_int} > {0}
      {
        \int_compare:nNnT {\l__kanban_deadline_month_int} > {0}
          {
            \int_compare:nNnT {\l__kanban_deadline_month_int} < {13}
              {
                \__kanban_set_max_day:
                \int_compare:nNnT {\l__kanban_deadline_day_int} > {0}
                  {
                    \int_compare:nNnT
                      {\l__kanban_deadline_day_int}
                      < {\l__kanban_deadline_max_day_int + 1}
                      {\bool_set_true:N \l__kanban_valid_date_bool}
                  }
              }
          }
      }
  }
\cs_new_protected:Npn \__kanban_set_match:Nn #1#2
  {
    \int_set:Nn #1
      {\seq_item:Nn \l__kanban_deadline_match_seq {#2}}
  }
\cs_new_protected:Npn \__kanban_set_components:nnn #1#2#3
  {
    \bool_set_true:N \l__kanban_date_syntax_bool
    \bool_set_true:N \l__kanban_date_components_bool
    \__kanban_set_match:Nn \l__kanban_deadline_year_int {#1}
    \__kanban_set_match:Nn \l__kanban_deadline_month_int {#2}
    \__kanban_set_match:Nn \l__kanban_deadline_day_int {#3}
  }
```

Recognition of the five date notations. The parser recognizes dotted day-first dates, hyphenated ISO dates, slash-separated year-first dates, and the two locale-dependent slash forms. Every notation requires a four-digit year. Month and day may have one or two digits. For an ambiguous year-last slash date, the selected date format determines the first interpretation. If that interpretation is impossible, the parser swaps day and month once and validates the alternative, which accepts unambiguous input such as `31/12/2026` under any format.

Input that has the shape of a date but matches none of the exact notations, for example a two-digit year, sets only the syntax flag. Such input is reported as an invalid date and is never silently treated as a label. This protects against `18.07.26` being read as a date in the year 26.

The input is expanded through `\text_expand:n` before matching. Ordinary macros are expanded to their text, so a macro that holds a date is classified. Robust and protected commands are left intact, so a deadline such as `\emph{soon}` is a neutral label and cannot derail the expanded assignment.

```
\cs_new_protected:Npn \__kanban_parse_date:nn #1#2
  {
    \bool_set_false:N \l__kanban_valid_date_bool
    \bool_set_false:N \l__kanban_date_syntax_bool
    \bool_set_false:N \l__kanban_date_components_bool
    \bool_set_false:N \l__kanban_ambiguous_slash_bool
```

```
575     \tl_set:Nx \l__kanban_deadline_tl {\text_expand:n {#1}}
576     \tl_trim_spaces:N \l__kanban_deadline_tl
577     \exp_args:NnV \regex_extract_once:nnNTF
578       {\A([0-9]{1,2})\.([0-9]{1,2})\.([0-9]{4})\Z}
579       \l__kanban_deadline_tl
580       \l__kanban_deadline_match_seq
581       {\__kanban_set_components:nnn {4}{3}{2}}
582       {
583         \exp_args:NnV \regex_extract_once:nnNTF
584           {\A([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})\Z}
585           \l__kanban_deadline_tl
586           \l__kanban_deadline_match_seq
587           {\__kanban_set_components:nnn {2}{3}{4}}
588           {
589             \exp_args:NnV \regex_extract_once:nnNTF
590               {\A([0-9]{4})/([0-9]{1,2})/([0-9]{1,2})\Z}
591               \l__kanban_deadline_tl
592               \l__kanban_deadline_match_seq
593               {\__kanban_set_components:nnn {2}{3}{4}}
594               {
595                 \exp_args:NnV \regex_extract_once:nnNTF
596                   {\A([0-9]{1,2})/([0-9]{1,2})/([0-9]{4})\Z}
597                   \l__kanban_deadline_tl
598                   \l__kanban_deadline_match_seq
599                   {
600                     \bool_set_true:N \l__kanban_ambiguous_slash_bool
601                     \bool_if:nTF
602                       {
603                         \str_if_eq_p:ee {#2} {english}
604                         || \str_if_eq_p:ee {#2} {german}
605                       }
606                       {\__kanban_set_components:nnn {4}{3}{2}}
607                       {\__kanban_set_components:nnn {4}{2}{3}}
608                   }
609                   {
610                     \exp_args:NnV \regex_match:nnT
611                       {\A[0-9]{1,4}[.\-/][0-9]{1,2}[.\-/][0-9]{1,4}\Z}
612                       \l__kanban_deadline_tl
613                       {\bool_set_true:N \l__kanban_date_syntax_bool}
614                   }
615               }
616           }
617       }
618     \bool_if:NT \l__kanban_date_components_bool
619       {\__kanban_validate_date:}
620     \bool_if:nT
621       {
622         \l__kanban_ambiguous_slash_bool
623         && ! \l__kanban_valid_date_bool
624       }
625       {
626         \int_set_eq:NN \l__kanban_swap_int \l__kanban_deadline_day_int
627         \int_set_eq:NN \l__kanban_deadline_day_int
628           \l__kanban_deadline_month_int
629         \int_set_eq:NN \l__kanban_deadline_month_int \l__kanban_swap_int
630         \__kanban_validate_date:
631       }
632   }
```

Canonical and display formatting. Accepted dates are stored internally in one unambiguous year, month, and day representation. At rendering time the same numeric components are formatted according to the notation selected for the board. Keeping canonical storage and presentation separate ensures that `date-format` affects input interpretation and display but never chronological comparison. The padding helpers are expandable so that the canonical form can be produced inside an expanded definition. The display macro keeps the literal deadline for invalid dates and descriptive labels, so those values remain printable.

```
\cs_new:Npn \__kanban_two_digits:n #1
  {
    \int_compare:nNnT {#1} < {10} {0}
    \int_to_arabic:n {#1}
  }
\cs_new:Npn \__kanban_four_digits:n #1
  {
    \int_compare:nNnT {#1} < {1000} {0}
    \int_compare:nNnT {#1} < {100} {0}
    \int_compare:nNnT {#1} < {10} {0}
    \int_to_arabic:n {#1}
  }
\cs_new:Npn \__kanban_canonical_date:
  {
    \__kanban_four_digits:n {\l__kanban_deadline_year_int}
    -\__kanban_two_digits:n {\l__kanban_deadline_month_int}
    -\__kanban_two_digits:n {\l__kanban_deadline_day_int}
  }
\cs_new_protected:Npn \kanban@preparedeadline #1
  {
    \cs_set:Npn \kb@displaydeadline {#1}
    \__kanban_parse_date:nn {#1} {\kanban@boarddateformat}
    \bool_if:NT \l__kanban_valid_date_bool
      {
        \tl_set:Nx \l__kanban_display_day_tl
          {\__kanban_two_digits:n {\l__kanban_deadline_day_int}}
        \tl_set:Nx \l__kanban_display_month_tl
          {\__kanban_two_digits:n {\l__kanban_deadline_month_int}}
        \tl_set:Nx \l__kanban_display_year_tl
          {\__kanban_four_digits:n {\l__kanban_deadline_year_int}}
        \tl_set:Nx \l__kanban_display_format_tl
          {\kanban@boarddateformat}
        \str_case:VnF \l__kanban_display_format_tl
          {
            {us}
              {
                \cs_set:Npx \kb@displaydeadline
                  {
                    \l__kanban_display_month_tl /
                    \l__kanban_display_day_tl /
                    \l__kanban_display_year_tl
                  }
              }
            {english}
              {
                \cs_set:Npx \kb@displaydeadline
                  {
                    \l__kanban_display_day_tl /
                    \l__kanban_display_month_tl /
                    \l__kanban_display_year_tl
```

```
                }
            }
          {german}
            {
              \cs_set:Npx \kb@displaydeadline
                {
                  \l__kanban_display_day_tl .
                  \l__kanban_display_month_tl .
                  \l__kanban_display_year_tl
                }
            }
          {iso}
            {
              \cs_set:Npx \kb@displaydeadline
                {
                  \l__kanban_display_year_tl -
                  \l__kanban_display_month_tl -
                  \l__kanban_display_day_tl
                }
            }
          {ymd}
            {
              \cs_set:Npx \kb@displaydeadline
                {
                  \l__kanban_display_year_tl /
                  \l__kanban_display_month_tl /
                  \l__kanban_display_day_tl
                }
            }
        }
        { }
    }
}
```

Project-month syntax. Project deadlines such as `M1` are relative to the configured start date rather than to a calendar date written on the card. The parser accepts a positive integer of at most six significant digits and allows leading zeros, so `M01` denotes the first month while `M0` and `M000` remain invalid. It records recognized but invalid `M`-prefixed numeric syntax separately and leaves all other text neutral. The bound keeps malformed input away from integer arithmetic while remaining far beyond any realistic project duration. The input is expanded through `\text_expand:n` in the same way as a calendar date.

```
\cs_new_protected:Npn \__kanban_parse_project_month:n #1
  {
    \bool_set_false:N \l__kanban_valid_project_month_bool
    \bool_set_false:N \l__kanban_project_month_syntax_bool
    \tl_set:Nx \l__kanban_deadline_tl {\text_expand:n {#1}}
    \tl_trim_spaces:N \l__kanban_deadline_tl
    \exp_args:NnV \regex_match:nnT
      {\AM[0-9]+\Z}
      \l__kanban_deadline_tl
      {\bool_set_true:N \l__kanban_project_month_syntax_bool}
    \exp_args:NnV \regex_extract_once:nnNTF
      {\AM0*([1-9][0-9]{0,5})\Z}
      \l__kanban_deadline_tl
      \l__kanban_deadline_match_seq
```

```
730       {
731         \int_set:Nn \l__kanban_project_month_int
732           {\seq_item:Nn \l__kanban_deadline_match_seq {2}}
733         \bool_set_true:N \l__kanban_valid_project_month_bool
734       }
735       { }
736   }
```

Reference-date loading and calendar classification. Classification is evaluated against the board-local reference date. If none was configured, the engine reads the compilation year, month, and day from TeX. A configured value is already canonical, so one exact expression extracts it. Calendar dates are then converted to order-preserving integers of the form $10000y + 100m + d$. A smaller value is overdue and an equal value is due now. The two `etoolbox` Booleans are the only facts the card renderer consumes.

```
737 \cs_new_protected:Npn \__kanban_load_reference_date:
738   {
739     \tl_set:Nx \l__kanban_reference_tl {\kanban@thereference}
740     \tl_if_blank:VTF \l__kanban_reference_tl
741       {
742         \int_set:Nn \l__kanban_reference_year_int {\year}
743         \int_set:Nn \l__kanban_reference_month_int {\month}
744         \int_set:Nn \l__kanban_reference_day_int {\day}
745       }
746       {
747         \exp_args:NnV \regex_extract_once:nnN
748           {\A([0-9]{4})-([0-9]{2})-([0-9]{2})\Z}
749           \l__kanban_reference_tl \l__kanban_deadline_match_seq
750         \int_set:Nn \l__kanban_reference_year_int
751           {\seq_item:Nn \l__kanban_deadline_match_seq {2}}
752         \int_set:Nn \l__kanban_reference_month_int
753           {\seq_item:Nn \l__kanban_deadline_match_seq {3}}
754         \int_set:Nn \l__kanban_reference_day_int
755           {\seq_item:Nn \l__kanban_deadline_match_seq {4}}
756       }
757   }
758 \cs_new_protected:Npn \__kanban_classify_calendar_date:
759   {
760     \__kanban_load_reference_date:
761     \int_set:Nn \l__kanban_deadline_key_int
762       {\l__kanban_deadline_year_int * 10000
763         + \l__kanban_deadline_month_int * 100
764         + \l__kanban_deadline_day_int}
765     \int_set:Nn \l__kanban_reference_key_int
766       {\l__kanban_reference_year_int * 10000
767         + \l__kanban_reference_month_int * 100
768         + \l__kanban_reference_day_int}
769     \int_compare:nNnTF
770       {\l__kanban_deadline_key_int} < {\l__kanban_reference_key_int}
771       {\booltrue{kb@overdue}}
772       {
773         \int_compare:nNnT
774           {\l__kanban_deadline_key_int} = {\l__kanban_reference_key_int}
775           {\booltrue{kb@duenow}}
776       }
777   }
```

Project-month classification. A project month begins on the configured start-day number, not necessarily on the first day of a calendar month. The classifier computes the nominal month offset from the project start, clamps the boundary day to the last valid day of a shorter reference month, and subtracts one when the reference date has not yet reached that boundary. The reference year and month are placed temporarily in the date component variables so that the shared maximum-day function can be reused.

```
\cs_new_protected:Npn \__kanban_classify_project_month:
  {
    \__kanban_load_reference_date:
    \int_set:Nn \l__kanban_current_month_int
      {(\l__kanban_reference_year_int
          - \l__kanban_project_start_year_int) * 12
        + (\l__kanban_reference_month_int
          - \l__kanban_project_start_month_int) + 1}
    \int_set_eq:NN \l__kanban_deadline_year_int
      \l__kanban_reference_year_int
    \int_set_eq:NN \l__kanban_deadline_month_int
      \l__kanban_reference_month_int
    \__kanban_set_max_day:
    \int_set_eq:NN \l__kanban_project_boundary_day_int
      \l__kanban_project_start_day_int
    \int_compare:nNnT
      {\l__kanban_project_boundary_day_int}
      > {\l__kanban_deadline_max_day_int}
      {\int_set_eq:NN \l__kanban_project_boundary_day_int
        \l__kanban_deadline_max_day_int}
    \int_compare:nNnT
      {\l__kanban_reference_day_int}
      < {\l__kanban_project_boundary_day_int}
      {\int_decr:N \l__kanban_current_month_int}
    \int_compare:nNnTF
      {\l__kanban_current_month_int}
      > {\l__kanban_project_month_int}
      {\booltrue{kb@overdue}}
      {
        \int_compare:nNnT
          {\l__kanban_current_month_int}
          = {\l__kanban_project_month_int}
          {\booltrue{kb@duenow}}
      }
  }
```

`\kb@evaloverdue` This internal entry point reduces every deadline value to two renderer-facing facts, overdue and due now. It resets both facts, tries the calendar grammar first, and otherwise distinguishes a project month from a neutral label. A project month is classified only when the board has a valid start-date snapshot. The function issues no diagnostics, because every invalid value has already been reported when its task was declared.

```
\cs_new_protected:Npn \kb@evaloverdue #1
  {
    \boolfalse{kb@overdue}
    \boolfalse{kb@duenow}
    \__kanban_parse_date:nn {#1} {\kanban@boarddateformat}
```

```
    \bool_if:NTF \l__kanban_valid_date_bool
      {\__kanban_classify_calendar_date:}
      {
        \bool_if:NF \l__kanban_date_syntax_bool
          {
            \__kanban_parse_project_month:n {#1}
            \bool_if:NT \l__kanban_valid_project_month_bool
              {
                \__kanban_parse_date:nn
                  {\kanban@thestart} {\kanban@boarddateformat}
                \bool_if:NT \l__kanban_valid_date_bool
                  {
                    \int_set_eq:NN \l__kanban_project_start_day_int
                      \l__kanban_deadline_day_int
                    \int_set_eq:NN \l__kanban_project_start_month_int
                      \l__kanban_deadline_month_int
                    \int_set_eq:NN \l__kanban_project_start_year_int
                      \l__kanban_deadline_year_int
                    \__kanban_classify_project_month:
                  }
              }
          }
      }
  }
```

Deadline diagnostics at declaration time. The collector calls this check for every accepted task that carries a deadline. A value with date syntax that fails validation, and a value with project-month syntax that lies outside the accepted range, each produce one named warning. Because the check runs while the task declaration is being read, the warning carries the line number of the task. Descriptive labels produce no diagnostic.

```
\cs_new_protected:Npn \__kanban_deadline_diagnose:n #1
  {
    \__kanban_parse_date:nn {#1} {\kanban@boarddateformat}
    \bool_if:NF \l__kanban_valid_date_bool
      {
        \bool_if:NTF \l__kanban_date_syntax_bool
          {\msg_warning:nnn {kanban-board}{invalid-deadline}{#1}}
          {
            \__kanban_parse_project_month:n {#1}
            \bool_if:nT
              {
                \l__kanban_project_month_syntax_bool
                && ! \l__kanban_valid_project_month_bool
              }
              {\msg_warning:nnn {kanban-board}{invalid-project-month}{#1}}
          }
      }
  }
\cs_new_protected:Npn \kanban@diagnosedeadline #1
  {\tl_if_blank:nF {#1} {\__kanban_deadline_diagnose:n {#1}}}
```

Validated persistent and board-local date setters. Start and reference dates enter the package through several configuration scopes, but every accepted non-

empty value is stored canonically. The persistent setters update the defaults for future boards globally. The board setters update only the current environment. Empty values clear the corresponding setting. Invalid non-empty values keep the previously accepted value and issue the shared named warning. The expanded definitions store a self-contained canonical date rather than a reference to the mutable parsing integers.

```
\cs_new_protected:Npn \kanban@setstartdate #1
  {
    \tl_if_blank:nTF {#1}
      {\gdef\kanban@startdate{}}
      {
        \__kanban_parse_date:nn {#1} {\kanban@dateformat}
        \bool_if:NTF \l__kanban_valid_date_bool
          {\xdef\kanban@startdate{\__kanban_canonical_date:}}
          {\msg_warning:nnnn {kanban-board}{invalid-date-setting}
            {startdate}{#1}}
      }
  }
\cs_new_protected:Npn \kanban@setboardstartdate #1
  {
    \tl_if_blank:nTF {#1}
      {\def\kanban@thestart{}}
      {
        \__kanban_parse_date:nn {#1} {\kanban@boarddateformat}
        \bool_if:NTF \l__kanban_valid_date_bool
          {\edef\kanban@thestart{\__kanban_canonical_date:}}
          {\msg_warning:nnnn {kanban-board}{invalid-date-setting}
            {board~startdate}{#1}}
      }
  }
\cs_new_protected:Npn \kanban@setreferencedate #1
  {
    \tl_if_blank:nTF {#1}
      {\gdef\kanban@referencedate{}}
      {
        \__kanban_parse_date:nn {#1} {\kanban@dateformat}
        \bool_if:NTF \l__kanban_valid_date_bool
          {\xdef\kanban@referencedate{\__kanban_canonical_date:}}
          {\msg_warning:nnnn {kanban-board}{invalid-date-setting}
            {reference-date}{#1}}
      }
  }
\cs_new_protected:Npn \kanban@setboardreferencedate #1
  {
    \tl_if_blank:nTF {#1}
      {\def\kanban@thereference{}}
      {
        \__kanban_parse_date:nn {#1} {\kanban@boarddateformat}
        \bool_if:NTF \l__kanban_valid_date_bool
          {\edef\kanban@thereference{\__kanban_canonical_date:}}
          {\msg_warning:nnnn {kanban-board}{invalid-date-setting}
            {board~reference-date}{#1}}
      }
  }
```

Safe dimension recognition. The column-gap setters must reject malformed input before TeX attempts a dimension assignment, because an unrestricted assign-

ment can consume later document tokens while searching for a unit. The validator accepts three forms. The first is a complete explicit dimension in a standard TeX unit, with optional spaces before the unit and an optional `true` prefix on the physical units. The second is one existing dimension or skip register. The third is one token whose meaning is a dimension or glue parameter of TeX, such as `\parindent` or `\baselineskip`. Those parameters are primitives rather than registers, so the register tests do not recognize them. The constant lists the meaning of all 21 dimension parameters and all 15 glue parameters of TeX, and the membership test compares the meaning of the candidate token, which also accepts a `\let` copy of a parameter. The three math glue parameters are excluded because they cannot be assigned to a length. The validator does not evaluate arbitrary macros or dimension expressions at this boundary. The input is expanded through `\text_expand:n`, so a macro that expands to a dimension is accepted while a robust command is left intact and rejected. The two LaTeX2e-facing wrappers translate the conditional and the named message for the configuration layer.

```
\str_const:Nx \c__kanban_tex_lengths_str
  {
    | \token_to_meaning:N \parindent
    | \token_to_meaning:N \mathsurround
    | \token_to_meaning:N \lineskiplimit
    | \token_to_meaning:N \hsize
    | \token_to_meaning:N \vsize
    | \token_to_meaning:N \maxdepth
    | \token_to_meaning:N \splitmaxdepth
    | \token_to_meaning:N \boxmaxdepth
    | \token_to_meaning:N \hfuzz
    | \token_to_meaning:N \vfuzz
    | \token_to_meaning:N \delimitershortfall
    | \token_to_meaning:N \nulldelimiterspace
    | \token_to_meaning:N \scriptspace
    | \token_to_meaning:N \predisplaysize
    | \token_to_meaning:N \displaywidth
    | \token_to_meaning:N \displayindent
    | \token_to_meaning:N \overfullrule
    | \token_to_meaning:N \hangindent
    | \token_to_meaning:N \hoffset
    | \token_to_meaning:N \voffset
    | \token_to_meaning:N \emergencystretch
    | \token_to_meaning:N \lineskip
    | \token_to_meaning:N \baselineskip
    | \token_to_meaning:N \parskip
    | \token_to_meaning:N \abovedisplayskip
    | \token_to_meaning:N \abovedisplayshortskip
    | \token_to_meaning:N \belowdisplayskip
    | \token_to_meaning:N \belowdisplayshortskip
    | \token_to_meaning:N \leftskip
    | \token_to_meaning:N \rightskip
    | \token_to_meaning:N \topskip
    | \token_to_meaning:N \splittopskip
    | \token_to_meaning:N \tabskip
    | \token_to_meaning:N \spaceskip
    | \token_to_meaning:N \xspaceskip
    | \token_to_meaning:N \parfillskip |
  }
\prg_new_protected_conditional:Npnn
  \__kanban_dimension_token:N #1 {T,F,TF}
```

```
  {
    \token_if_dim_register:NTF #1
      {\prg_return_true:}
      {
        \token_if_skip_register:NTF #1
          {\prg_return_true:}
          {
            \exp_args:NNx \str_if_in:NnTF \c__kanban_tex_lengths_str
              { | \token_to_meaning:N #1 | }
              {\prg_return_true:}
              {\prg_return_false:}
          }
      }
  }
\prg_new_protected_conditional:Npnn
  \__kanban_dimension_valid:n #1 {T,F,TF}
  {
    \tl_set:Nx \l__kanban_dimension_tl {\text_expand:n {#1}}
    \tl_trim_spaces:N \l__kanban_dimension_tl
    \exp_args:NnV \regex_match:nnTF
      {\A[+\-]?(?:[0-9]+(?:\.[0-9]*)?|\.[0-9]+)\s*
        (?:(?:true\s*)?(?:pt|pc|in|bp|cm|mm|dd|cc|sp)|ex|em)\Z}
      \l__kanban_dimension_tl
      {\prg_return_true:}
      {
        \exp_args:NV \tl_if_single_token:nTF \l__kanban_dimension_tl
          {
            \exp_args:NV \__kanban_dimension_token:NTF
              \l__kanban_dimension_tl
              {\prg_return_true:}
              {\prg_return_false:}
          }
          {\prg_return_false:}
      }
  }
\cs_new_protected:Npn \kanban@validatedimensionTF #1#2#3
  {\__kanban_dimension_valid:nTF {#1}{#2}{#3}}
\cs_new_protected:Npn \kanban@dimensionwarning #1#2
  {\msg_warning:nnnn {kanban-board}{invalid-dimension}{#1}{#2}}
\ExplSyntaxOff
```

```
⟨/package⟩
```

```
⟨*package⟩
```

11.8 Persistent configuration

Built-in palettes and the default column list are registered through the same mechanisms that documents use. Mode, date format, start date, reference date, column list, palette, semantic labels, and gap defaults then enter the dedicated `kanban/setup` key family. Package options are accumulated and passed through the same transaction, so package-level and document-level setup have identical validation and scope. Throughout this part, a package option or `\kbset` changes the source for later snapshots, whereas an environment key changes only the snapshot

being constructed.

Built-in palettes and default columns. The default, muted, and monochrome palettes obey exactly the same validation and registry rules as user-defined palettes, because they are registered through the public command. The default palette is selected immediately so that a document with no configuration produces the documented colors. The default column list is installed through the persistent column selector for the same reason. Its spelling is the text shown in the headings, and its uppercase form is the lookup key.

```
\kanbandefinepalette{default}{%
  kanbanSlate,kanbanBlue,kanbanTeal,kanbanSage,
  kanbanAmber,kanbanClay,kanbanRose,kanbanPlum}
\kanbandefinepalette{muted}{kanbanMutedBlue,kanbanMutedGreen,
  kanbanMutedPurple,kanbanMutedBrown}
\kanbandefinepalette{monochrome}{black!85,black!75,black!65,black!55}
\kanban@selectpalette{default}
\kanban@selectcolumns{Backlog,Doing,Done}
```

Mode selection. Visual mode is a constrained textual choice. The persistent selector records a global `light` or `dark` default and the board selector records the same choice locally. Unknown values keep the previous selection and are reported through the shared unknown-value message, which names the scope of the setting.

```
\newcommand\kanban@selectmode[1]{%
  \kanban@normalizechoice{#1}%
  \ifdefstring{\kb@choice}{light}{\gdef\kanban@mode{light}}{%
    \ifdefstring{\kb@choice}{dark}{\gdef\kanban@mode{dark}}{%
      \kanban@warning{unknown-value}{mode}{#1}}}}
\newcommand\kanban@selectboardmode[1]{%
  \kanban@normalizechoice{#1}%
  \ifdefstring{\kb@choice}{light}{\def\kanban@boardmode{light}}{%
    \ifdefstring{\kb@choice}{dark}{\def\kanban@boardmode{dark}}{%
      \kanban@warning{unknown-value}{board mode}{#1}}}}
```

Date-format canonicalization and selection. Five canonical format names are supported together with a small set of readable aliases. Canonicalization is separated from assignment so that the same alias table serves persistent and board-local selection. A recognized value writes one of `us`, `english`, `german`, `iso`, or `ymd`. An unknown value leaves the result empty and is diagnosed by the selector.

```
\newcommand\kanban@canonicaldateformat[1]{%
  \kanban@normalizechoice{#1}%
  \def\kb@datechoice{}%
  \ifdefstring{\kb@choice}{us}{\def\kb@datechoice{us}}{}%
  \ifdefstring{\kb@choice}{american}{\def\kb@datechoice{us}}{}%
  \ifdefstring{\kb@choice}{english}{\def\kb@datechoice{english}}{}%
  \ifdefstring{\kb@choice}{uk}{\def\kb@datechoice{english}}{}%
  \ifdefstring{\kb@choice}{british}{\def\kb@datechoice{english}}{}%
  \ifdefstring{\kb@choice}{german}{\def\kb@datechoice{german}}{}%
  \ifdefstring{\kb@choice}{de}{\def\kb@datechoice{german}}{}%
```

```
  \ifdefstring{\kb@choice}{deutsch}{\def\kb@datechoice{german}}{}%
  \ifdefstring{\kb@choice}{iso}{\def\kb@datechoice{iso}}{}%
  \ifdefstring{\kb@choice}{ymd}{\def\kb@datechoice{ymd}}{}%
  \ifdefstring{\kb@choice}{year-first}{\def\kb@datechoice{ymd}}{}}
\newcommand\kanban@selectdateformat[1]{%
  \kanban@canonicaldateformat{#1}%
  \ifdefempty\kb@datechoice
    {\kanban@warning{unknown-value}{date format}{#1}}
    {\global\let\kanban@dateformat\kb@datechoice}}
\newcommand\kanban@selectboarddateformat[1]{%
  \kanban@canonicaldateformat{#1}%
  \ifdefempty\kb@datechoice
    {\kanban@warning{unknown-value}{board date format}{#1}}
    {\let\kanban@boarddateformat\kb@datechoice}}
```

Validated column-gap assignment. A column gap must be a non-negative dimension. The persistent and board-local setters share the safe grammar of the deadline part, assign the accepted input to a candidate register, and compare the evaluated length with zero before committing it to the appropriate scope.

```
\newcommand\kanban@setcolumnsep[1]{%
  \kanban@validatedimensionTF{#1}{%
    \setlength\kanban@candidatecolumnsep{#1}%
    \ifdim\kanban@candidatecolumnsep<0pt
      \kanban@dimensionwarning{columnsep}{#1}%
    \else
      \global\kanban@columnsep=\kanban@candidatecolumnsep
    \fi
  }{%
    \kanban@dimensionwarning{columnsep}{#1}}}
\newcommand\kanban@setboardcolumnsep[1]{%
  \kanban@validatedimensionTF{#1}{%
    \setlength\kanban@candidatecolumnsep{#1}%
    \ifdim\kanban@candidatecolumnsep<0pt
      \kanban@dimensionwarning{board columnsep}{#1}%
    \else
      \kanban@boardcolumnsep=\kanban@candidatecolumnsep
    \fi
  }{%
    \kanban@dimensionwarning{board columnsep}{#1}}}
```

Materialization of the visual mode. The mode name is semantic state. Rendering needs concrete text, surface, border, and default heading colors, which this function derives from the active board mode after all board options have been processed. The tint percentage used for optional card backgrounds also depends on the mode. A light surface needs only a touch of color, whereas a dark surface needs a stronger share to stay visible while keeping the text contrast. Metadata chip colors are absent here because their light styling is stable in both modes.

```
\newcommand\kanban@applymode{%
  \ifdefstring{\kanban@boardmode}{dark}{%
    \def\kanban@textcolor{white!88!black}%
    \def\kanban@cardcolor{kanbanInk}%
    \def\kanban@cardtint{20}%
```

```
    \def\kanban@bordercolor{white!18!black}%
    \def\kanban@headingtext{white}%
  }{%
    \def\kanban@textcolor{kanbanInk}%
    \def\kanban@cardcolor{white}%
    \def\kanban@cardtint{15}%
    \def\kanban@bordercolor{black!15}%
    \def\kanban@headingtext{white}%
  }}
```

Persistent and board-local semantic labels. Priority and completion labels are customizable text, but an empty label would create an invisible chip with misleading spacing. Both setters reject blank input. The label identifier supplied by the key family selects the persistent or board-local macro by name.

```
\newcommand\kanban@setlabel[2]{%
  \ifblank{#2}
    {\kanban@warning{empty-label}{label}{#1}}
    {\csgdef{kanban@#1label}{#2}}}
\newcommand\kanban@setboardlabel[2]{%
  \ifblank{#2}
    {\kanban@warning{empty-label}{board label}{#1}}
    {\csdef{kanban@board#1label}{#2}}}
```

Nested label key families. Label customization is expressed as a subordinate key list such as `labels={high=URGENT,done=FINISHED}`. Separate persistent and board-local families direct identical key names to setters with different assignment scope. Each family owns its unknown-key diagnostic, which states whether the mistake occurred in global setup or in a board override. The key name is passed by expansion, because l3msg shows message arguments without expanding them.

```
\ExplSyntaxOn
\keys_define:nn {kanban/setup-labels}
  {
    high .code:n = {\kanban@setlabel{high}{#1}},
    medium .code:n = {\kanban@setlabel{medium}{#1}},
    low .code:n = {\kanban@setlabel{low}{#1}},
    done .code:n = {\kanban@setlabel{done}{#1}},
    unknown .code:n =
      {\use:x {\msg_warning:nnnn {kanban-board}{unknown-key}
        {label}{\l_keys_key_str}}}
  }
\keys_define:nn {kanban/board-labels}
  {
    high .code:n = {\kanban@setboardlabel{high}{#1}},
    medium .code:n = {\kanban@setboardlabel{medium}{#1}},
    low .code:n = {\kanban@setboardlabel{low}{#1}},
    done .code:n = {\kanban@setboardlabel{done}{#1}},
    unknown .code:n =
      {\use:x {\msg_warning:nnnn {kanban-board}{unknown-key}
        {board~label}{\l_keys_key_str}}}
  }
```

The persistent setup key family. The setup family is the single entry point for package options and for `\kanbansetup`. Immediate keys delegate to their validated selectors. Date values are staged rather than applied at once, because a `date-format` key appearing later in the same list must take effect before an ambiguous start or reference date is interpreted. Alias keys provide the compact and the legacy spellings without a second implementation.

```
\keys_define:nn {kanban/setup}
  {
    columns .code:n = {\kanban@selectcolumns{#1}},
    palette .code:n = {\kanban@selectpalette{#1}},
    date-format .code:n = {\kanban@selectdateformat{#1}},
    dateformat .meta:n = {date-format={#1}},
    startdate .code:n =
      {
        \tl_set:Nn \l__kanban_pending_startdate_tl {#1}
        \bool_set_true:N \l__kanban_pending_startdate_bool
      },
    reference-date .code:n =
      {
        \tl_set:Nn \l__kanban_pending_reference_date_tl {#1}
        \bool_set_true:N \l__kanban_pending_reference_date_bool
      },
    today .meta:n = {reference-date={#1}},
    mode .code:n = {\kanban@selectmode{#1}},
    columnsep .code:n = {\kanban@setcolumnsep{#1}},
    labels .code:n = {\keys_set:nn {kanban/setup-labels}{#1}},
    unknown .code:n =
      {\use:x {\msg_warning:nnnn {kanban-board}{unknown-key}
        {setup}{\l_keys_key_str}}}
  }
```

A deferred setup transaction. Processing a setup list is a small transaction. It clears both staging flags, processes the complete list in source order, and only then applies any staged dates under the final selected format. Consequently `startdate=07/08/2026,date-format=english` and the reversed key order have the same interpretation.

```
\cs_new_protected:Npn \kanban@processsetup #1
  {
    \bool_set_false:N \l__kanban_pending_startdate_bool
    \bool_set_false:N \l__kanban_pending_reference_date_bool
    \keys_set:nn {kanban/setup}{#1}
    \bool_if:NT \l__kanban_pending_startdate_bool
      {\exp_args:NV \kanban@setstartdate
        \l__kanban_pending_startdate_tl}
    \bool_if:NT \l__kanban_pending_reference_date_bool
      {\exp_args:NV \kanban@setreferencedate
        \l__kanban_pending_reference_date_tl}
  }
\ExplSyntaxOff
```

`\kanbansetup` `\kbset` The canonical setup command exposes the transaction with one mandatory key-list argument. The short alias follows the same collision-tolerant policy as the palette

alias. Equivalence by `\let` gives both names exactly the same argument parser and behavior.

```
\NewDocumentCommand\kanbansetup{m}{\kanban@processsetup{#1}}
\@ifundefined{kbset}
  {\let\kbset\kanbansetup}
  {\kanban@warning{alias-taken}{kbset}{kanbansetup}}
```

Collection and application of package options. LaTeX2e presents package options one at a time. The package collects them into one token list and applies that list through the deferred setup transaction after option processing has finished. Package options therefore share validation, aliases, key-order independence, and diagnostics with `\kanbansetup`. The transaction runs while `kanban-board.sty` is still being read, so the line number at that moment points into the package rather than into the document. The application therefore runs in a group that redefines the line context of l3msg to name the package options. Every assignment made by the setup transaction is global, so the group does not shorten the lifetime of any setting.

Formats dated 2021-06-01 or later parse raw option lists with the brace-aware key–value processor, so the catch-all collector receives every option intact, including values with embedded commas such as `columns={A,B}`. The 2020-10-01 format splits raw option lists at every comma and mishandles brace groups in its option bookkeeping. On such formats the package routes its options through kvoptions. Every documented setup key is declared as a keyval key whose handler re-wraps the brace-stripped value before appending it to the same accumulator, and unrecognized options fall through to the same catch-all collector. Unknown package options consequently produce the same warning as unknown setup keys on both paths.

```
\ExplSyntaxOn
\tl_new:N \g__kanban_package_options_tl
\cs_new_protected:Npn \kanban@collectpackageoption
  {
    \tl_if_empty:NF \g__kanban_package_options_tl
      {\tl_gput_right:Nn \g__kanban_package_options_tl {,}}
    \exp_args:NNo \tl_gput_right:Nn
      \g__kanban_package_options_tl {\CurrentOption}
  }
\cs_new_protected:Npn \kanban@collectkeyvalue #1#2
  {
    \tl_if_empty:NF \g__kanban_package_options_tl
      {\tl_gput_right:Nn \g__kanban_package_options_tl {,}}
    \tl_gput_right:Nn \g__kanban_package_options_tl {#1={#2}}
  }
\cs_new_protected:Npn \kanban@applypackageoptions
  {
    \tl_if_empty:NF \g__kanban_package_options_tl
      {
        \group_begin:
          \cs_set:Npn \msg_line_context: {in~the~package~options}
          \exp_args:NV \kanban@processsetup
            \g__kanban_package_options_tl
        \group_end:
      }
```

```
  }
\ExplSyntaxOff
\IfFormatAtLeastTF{2021-06-01}
  {%
    \DeclareOption*{\kanban@collectpackageoption}%
    \ProcessOptions\relax
  }
  {%
    \RequirePackage{kvoptions}%
    \SetupKeyvalOptions{family=kanban,prefix=kanban@pkgopt@}%
    \DeclareDefaultOption{\kanban@collectpackageoption}%
    \define@key{kanban}{columns}%
      {\kanban@collectkeyvalue{columns}{#1}}%
    \define@key{kanban}{palette}%
      {\kanban@collectkeyvalue{palette}{#1}}%
    \define@key{kanban}{date-format}%
      {\kanban@collectkeyvalue{date-format}{#1}}%
    \define@key{kanban}{dateformat}%
      {\kanban@collectkeyvalue{dateformat}{#1}}%
    \define@key{kanban}{startdate}%
      {\kanban@collectkeyvalue{startdate}{#1}}%
    \define@key{kanban}{reference-date}%
      {\kanban@collectkeyvalue{reference-date}{#1}}%
    \define@key{kanban}{today}%
      {\kanban@collectkeyvalue{today}{#1}}%
    \define@key{kanban}{mode}%
      {\kanban@collectkeyvalue{mode}{#1}}%
    \define@key{kanban}{columnsep}%
      {\kanban@collectkeyvalue{columnsep}{#1}}%
    \define@key{kanban}{labels}%
      {\kanban@collectkeyvalue{labels}{#1}}%
    \ProcessKeyvalOptions*\relax
  }
\kanban@applypackageoptions
```

```
⟨/package⟩
```

```
⟨*package⟩
```

11.9 Card components and renderers

Metadata elements are small inline boxes. Their semantic colors are stable across modes. Only the card surface, card text, and card border respond to the light or dark board mode. Public namespaced styles are appended after the structural defaults. Fixed and breakable layouts reuse the same card renderer but provide different column containers. Rendering is delayed. Task declarations store calls to these components, and the environment executes those calls only after it knows all column counts and dimensions.

Structural and extension styles. The style registry distinguishes fixed package structure from supported user extension points. The empty public styles may be extended with `\tcbset`. The private styles establish the common geometry of inline

pills and the stable author and deadline colors. Because the public extension key is applied last in each component definition, a document may override presentation without replacing structural collection logic.

```
\tcbset{
  kanban/header/.style={},
  kanban/card/.style={},
  kanban/column-body/.style={},
  kanban/priority-chip/.style={},
  kanban/author-chip/.style={},
  kanban/deadline-chip/.style={},
  kanban/completed-chip/.style={},
  kanban@pill/.style={on line,boxsep=0pt,left=1.2mm,right=1.2mm,
    top=.4mm,bottom=.4mm,arc=.7mm,boxrule=0pt,nobeforeafter,
    box align=base},
  kanban@chip/.style={kanban@pill,colback=\kanban@authorback,
    coltext=\kanban@authortext},
  kanban@due/.style={kanban@pill,colback=\kanban@deadlineback,
    coltext=\kanban@deadlinetext}}
```

Priority and author chips. A priority chip maps a normalized semantic value to the board-local label, the semantic background, and the matching foreground. An empty or unknown value produces no chip. The author component has no semantic classification and renders any non-blank author text with the stable author style.

```
\newcommand\kanban@prioritypill[1]{\ifblank{#1}{}{%
  \def\kb@pl{}\def\kb@pc{}\def\kb@pt{}%
  \ifstrequal{#1}{high}{\def\kb@pl{\kanban@boardhighlabel}%
    \def\kb@pc{kanbanHigh}\def\kb@pt{\kanban@hightext}}{}%
  \ifstrequal{#1}{medium}{\def\kb@pl{\kanban@boardmediumlabel}%
    \def\kb@pc{kanbanMed}\def\kb@pt{\kanban@mediumtext}}{}%
  \ifstrequal{#1}{med}{\def\kb@pl{\kanban@boardmediumlabel}%
    \def\kb@pc{kanbanMed}\def\kb@pt{\kanban@mediumtext}}{}%
  \ifstrequal{#1}{low}{\def\kb@pl{\kanban@boardlowlabel}%
    \def\kb@pc{kanbanLow}\def\kb@pt{\kanban@lowtext}}{}%
  \ifdefempty\kb@pl{}{\tcbox[kanban@pill,
    colback=\kb@pc,colframe=\kb@pc,coltext=\kb@pt,
    kanban/priority-chip]{\bfseries\scriptsize\kb@pl}}}}
\newcommand\kanban@authorchip[1]{\ifblank{#1}{}{%
  \tcbox[kanban@chip,kanban/author-chip]{\scriptsize #1}}}
```

The deadline clock symbol. Deadline chips include a compact clock drawn with TikZ primitives, so the package does not depend on an icon font. Its dimensions are fixed in points and small enough to align with script-sized metadata text. The picture is self-contained and does not alter later TikZ material.

```
\newcommand\kb@clock{\tikz[baseline=-.55ex]{%
  \draw[line width=.3pt] (0,0) circle (1.7pt);
  \draw[line width=.3pt] (0,0)--(0,1.1pt);
  \draw[line width=.3pt] (0,0)--(1pt,.2pt);}}
```

Deadline and completion chips. Deadline rendering first classifies the value and then prepares its display notation. A task marked `late` or an automatically overdue date receives the high semantic color. A date equal to the reference date receives the medium color. Future dates, project labels, and descriptive text keep the neutral deadline style. Completion is a separate flag rendered with the low semantic color. These colors are identical in light and dark board modes.

```
\newcommand\kanban@deadlinechip[2]{\ifblank{#2}{}{%
  \kb@evaloverdue{#2}%
  \kanban@preparedeadline{#2}%
  \ifboolexpr{test {\ifstrequal{#1}{1}} or bool {kb@overdue}}
    {\tcbox[kanban@due,colback=kanbanHigh,coltext=\kanban@hightext,
        kanban/deadline-chip]
      {\kb@clock\,\scriptsize\bfseries \kb@displaydeadline}}
    {\ifbool{kb@duenow}
      {\tcbox[kanban@due,colback=kanbanMed,coltext=\kanban@mediumtext,
          kanban/deadline-chip]
        {\kb@clock\,\scriptsize\bfseries \kb@displaydeadline}}
      {\tcbox[kanban@due,kanban/deadline-chip]
        {\kb@clock\,\scriptsize \kb@displaydeadline}}}}}
\newcommand\kanban@completedchip[1]{\ifstrequal{#1}{1}
  {\tcbox[kanban@pill,colback=kanbanLow,colframe=kanbanLow,
      coltext=\kanban@lowtext,kanban/completed-chip]
    {\bfseries\scriptsize\kanban@boarddonelabel}}{}}
```

Priority-value validation. Task options must store only priority values understood by the renderer. The setter normalizes case and surrounding spaces, accepts the four documented spellings, treats a blank choice as omission, and warns when non-blank input is unknown. The normalized value is stored with the task. All definitions occur while a task is collected inside the environment group, so the temporary value cannot leak to later tasks. The warning is a named message, so the definition sits in an expl3 syntax region.

The background setter validates the optional card tint before TeX reaches xcolor, because an unknown color would otherwise stop the run while the board is rendered, far from the declaration that caused it. The accepted grammar is a color name, optionally negated, followed by any number of `!`⟨*percentage*⟩`!`⟨*name*⟩ mixing steps. Every name in the expression must be known to xcolor. The value is expanded with `\text_expand:n` first, so an ordinary macro that holds a color name works and a robust command is rejected as unknown.

```
\ExplSyntaxOn
\cs_new_protected:Npn \kanban@setbackground #1
  {
    \tl_set:Nx \l__kanban_background_tl {\text_expand:n {#1}}
    \tl_trim_spaces:N \l__kanban_background_tl
    \cs_set:Npn \kbtask@bg {}
    \tl_if_blank:VF \l__kanban_background_tl
      {
        \bool_set_false:N \l__kanban_background_ok_bool
        \exp_args:NnV \regex_match:nnT
          {
            \A-?[A-Za-z][A-Za-z0-9]*
            (?:![0-9]{1,3}(?:![A-Za-z][A-Za-z0-9]*)?)*\Z
```

```
          }
          \l__kanban_background_tl
          {
            \bool_set_true:N \l__kanban_background_ok_bool
            \exp_args:NnV \regex_extract_all:nnN {[A-Za-z][A-Za-z0-9]*}
              \l__kanban_background_tl \l__kanban_background_seq
            \seq_map_inline:Nn \l__kanban_background_seq
              {
                \@ifundefinedcolor{##1}
                  {\bool_set_false:N \l__kanban_background_ok_bool}
                  {}
              }
          }
        \bool_if:NTF \l__kanban_background_ok_bool
          {\cs_set:Npx \kbtask@bg {\tl_use:N \l__kanban_background_tl}}
          {
            \use:x
              {
                \msg_warning:nnn {kanban-board}{unknown-color}
                  {\tl_to_str:N \l__kanban_background_tl}
              }
          }
      }
  }
\newcommand\kanban@setpriority[1]{%
  \kanban@normalizechoice{#1}%
  \def\kbtask@prio{}%
  \ifdefstring{\kb@choice}{high}{\def\kbtask@prio{high}}{%
    \ifdefstring{\kb@choice}{medium}{\def\kbtask@prio{medium}}{%
      \ifdefstring{\kb@choice}{med}{\def\kbtask@prio{med}}{%
        \ifdefstring{\kb@choice}{low}{\def\kbtask@prio{low}}{%
          \ifdefempty\kb@choice{}
            {\msg_warning:nnn {kanban-board}{unknown-priority}{#1}}}}}}}
\ExplSyntaxOff
```

Break-layout headers and shared task cards. The page-breakable renderer needs a header box whose height is synchronized across its independent columns through a board-specific equal-height group. All renderers need one full-width card whose surface and text follow the active mode and whose colored west rule identifies the target column. The card surface is either the plain surface of the mode or that surface mixed with a tint. The tint is a stored xcolor expression, so it is first bound to a named color and then mixed with the mode percentage, which keeps arbitrary expressions valid as a mixing operand. The shared card composer resolves the surface, places the task text first, then an optional row with priority, completion, and author, and finally an optional deadline row. Storing the same composer for both layouts guarantees identical card semantics.

```
\newtcolorbox{kanban@header}[2]{enhanced,sharp corners,colback=#1,
  boxrule=0pt,coltext=#2,
  fontupper=\bfseries\raggedright,
  left=2mm,right=2mm,
  top=1.2mm,bottom=1.2mm,before skip=0pt,after skip=1.8mm,
  equal height group={KBH@\kanban@boardgroup},valign=center,
  kanban/header}
\newtcolorbox{kanban@card}[2]{enhanced,arc=.6mm,outer arc=.6mm,
  colback=#2,colframe=\kanban@bordercolor,
```

```
  coltext=\kanban@textcolor,boxrule=.4pt,width=\linewidth,
  borderline west={2.2pt}{0pt}{#1},left=2.4mm,right=2mm,top=1.6mm,
  bottom=1.6mm,before skip=0pt,after skip=1.8mm,
  fontupper=\scriptsize\raggedright,kanban/card}
\newcommand\kanban@resolvecardback[1]{%
  \ifblank{#1}
    {\let\kb@cardback\kanban@cardcolor}
    {\colorlet{kb@tintbase}{#1}%
     \edef\kb@cardback{kb@tintbase!\kanban@cardtint!\kanban@cardcolor}}}
\newcommand\kanban@rendercard[8]{%
  \kanban@resolvecardback{#8}%
  \begin{kanban@card}{#1}{\kb@cardback}#2%
    \notblank{#3#4#7}{\par\vspace{1mm}\noindent
      \kanban@prioritypill{#3}%
      \notblank{#3}{\notblank{#7}{\hspace{1mm}}{}}{}%
      \kanban@completedchip{#7}\hfill\kanban@authorchip{#4}}{}%
    \notblank{#5}{\par\vspace{1mm}\noindent
      \kanban@deadlinechip{#6}{#5}}{}%
  \end{kanban@card}}
```

Fixed-board containers and renderer adapters. The fixed renderer uses a two-row raster. The first row contains synchronized headings and the second contains equal-height column bodies. The body hides both frame and interior, so the page shows through without a transparency group in the PDF, and draws only a bottom rule in a tint of the column color. The breakable renderer instead places each header and task list in a separate `paracol` stream. Three adapters insert the stored column name, count, color, foreground, and deferred card list into the appropriate container.

```
\newtcolorbox{kanban@fixedheader}[2]{enhanced,sharp corners,
  colback=#1,boxrule=0pt,coltext=#2,
  fontupper=\bfseries\raggedright,boxsep=0pt,left=2mm,right=2mm,
  top=1mm,bottom=1mm,before skip=0pt,after skip=0pt,valign=center,
  kanban/header}
\newtcolorbox{kanban@fixedbody}[1]{enhanced,sharp corners,frame hidden,
  interior hidden,boxrule=0pt,
  borderline south={.6pt}{0pt}{#1!55},boxsep=0pt,
  left=0pt,right=0pt,top=2mm,bottom=2mm,
  before skip=0pt,after skip=0pt,valign=top,kanban/column-body}
\newcommand\kanban@onehead[1]{%
  \begin{kanban@fixedheader}{\csuse{kanban@color@#1}}%
      {\csuse{kanban@headingtext@#1}}%
    \csuse{kanban@name@#1}\hfill
      \csuse{kanban@count@\kanban@boardgroup @#1}%
  \end{kanban@fixedheader}}
\newcommand\kanban@onecol[1]{%
  \begin{kanban@fixedbody}{\csuse{kanban@color@#1}}%
    \csuse{kanban@list@\kanban@boardgroup @#1}%
  \end{kanban@fixedbody}}
\newcommand\kanban@onecol@b[1]{%
  \begin{kanban@header}{\csuse{kanban@color@#1}}%
      {\csuse{kanban@headingtext@#1}}%
    \csuse{kanban@name@#1}\hfill
      \csuse{kanban@count@\kanban@boardgroup @#1}%
  \end{kanban@header}%
  \csuse{kanban@list@\kanban@boardgroup @#1}\par}
```

```
⟨/package⟩
```

```
⟨*package⟩
```

11.10 Task collection

The task key family stores metadata without expanding the card text. The key name of an unknown option is passed by expansion, because l3msg shows message arguments without expanding them. A valid declaration appends one protected renderer call to the selected board-local list. Invalid or missing targets are diagnosed and never affect the counter. Collection and rendering are separate phases. This permits the environment to determine the tallest heading, the highest column, and the final column counts before any task card is typeset.

Task-option keys. The task family translates user-facing metadata into temporary task macros. Priority and background use validated setters. Author and deadline keep their token lists. The `due` key aliases `deadline` and `background` aliases `bg`. The `late` and `completed` keys are valueless flags, so a misleading form such as `completed=false` is rejected. An unknown task option is ignored with a scoped warning.

```
\ExplSyntaxOn
\keys_define:nn {kanban/task}
  {
    prio .code:n = {\kanban@setpriority{#1}},
    author .code:n = {\def\kbtask@author{#1}},
    deadline .code:n = {\def\kbtask@deadline{#1}},
    due .meta:n = {deadline={#1}},
    late .code:n = {\def\kbtask@late{1}},
    late .value_forbidden:n = true,
    completed .code:n = {\def\kbtask@completed{1}},
    completed .value_forbidden:n = true,
    bg .code:n = {\kanban@setbackground{#1}},
    background .meta:n = {bg={#1}},
    unknown .code:n =
      {\use:x {\msg_warning:nnnn {kanban-board}{unknown-key}
        {task}{\l_keys_key_str}}}
  }
\cs_new_protected:Npn \kanban@processtaskoptions #1
  {\keys_set:nn {kanban/task}{#1}}
\ExplSyntaxOff
```

Validation and deferred task collection. The collector first gives every metadata field an empty value and processes the option list. It then requires an active board, a non-empty normalized target, and a target present in the identifier map of the board. Each failed condition emits a specific error and ignores the task. A task inside a rejected nested board is discarded silently because the nesting error has already explained the cause. A task outside any board is diagnosed on its own.

For a valid target, the counter is incremented globally and a self-contained eight-argument call to `\kanban@rendercard` is appended globally to the task list of the column. The renderer command is kept unexpanded for the rendering phase, the column color is resolved at once because it is already fixed, and the task body and all metadata are protected from expansion. The deadline is then checked once, so that any warning names the line of the declaration.

```
\newcommand\kanban@dotask[3][]{%
  \def\kbtask@prio{}\def\kbtask@author{}\def\kbtask@deadline{}%
  \def\kbtask@late{}\def\kbtask@completed{}\def\kbtask@bg{}%
  \kanban@processtaskoptions{#1}%
  \ifbool{kanban@active}
    {\kanban@normtaskcol{#2}%
     \ifdefempty\kbtask@column
       {\PackageError{kanban-board}
         {task has no assigned column.
           \MessageBreak Valid columns: \kanban@availablecolumns}
         {The first mandatory argument of
           \string\kanbantask\space must name a column from the
           enclosing board.\MessageBreak The task was ignored.}}
       {\ifdefempty\kbtask@columnid
       {\PackageError{kanban-board}
         {unknown kanban column '#2'.
           \MessageBreak Valid columns: \kanban@availablecolumns}
         {Column names are matched without regard to letter case.
           \MessageBreak
           The task was ignored because its normalized target is not in
           this board.}}
       {\expandafter\numgdef
           \csname kanban@count@\kanban@boardgroup
             @\kbtask@columnid\endcsname
           {\csname kanban@count@\kanban@boardgroup
             @\kbtask@columnid\endcsname+1}%
         \csxappto{kanban@list@\kanban@boardgroup
           @\kbtask@columnid}{%
           \noexpand\kanban@rendercard
           {\csuse{kanban@color@\kbtask@columnid}}{\unexpanded{#3}}%
           {\unexpanded\expandafter{\kbtask@prio}}%
           {\unexpanded\expandafter{\kbtask@author}}%
           {\unexpanded\expandafter{\kbtask@deadline}}%
           {\unexpanded\expandafter{\kbtask@late}}%
           {\unexpanded\expandafter{\kbtask@completed}}%
           {\unexpanded\expandafter{\kbtask@bg}}}%
         \expandafter\kanban@diagnosedeadline\expandafter{%
           \kbtask@deadline}%
       }}}
    {\ifbool{kanban@discardtasks}{}
      {\PackageError{kanban-board}
        {task declared outside a kanban environment}
        {Move the task declaration between \string\begin{kanban} and
          \string\end{kanban}. The task was ignored.}}}}
```

`\kanbantask` `\kbtask` The canonical public command supplies the compact task declaration syntax and delegates all key processing, validation, and collection to the internal implementation. The text argument is long, so a card may contain several paragraphs. The option list and the column name remain short, because a paragraph break inside them indicates a missing brace, and a short argument turns that mistake into an immediate error

instead of swallowing the rest of the document. The concise command is installed by equivalence only when its name is free. An existing definition is preserved with a warning.

```
\NewDocumentCommand\kanbantask{O{}m+m}{%
  \kanban@dotask[#1]{#2}{#3}}
\@ifundefined{kbtask}
  {\let\kbtask\kanbantask}
  {\kanban@warning{alias-taken}{kbtask}{kanbantask}}
```

11.11 Board lifecycle

Board keys update only the active board snapshot. Column setup allocates the isolated lists, prepares both rendering sequences, and validates the complete gap width before typesetting begins. The entry routine rejects nesting before any outer state can be replaced. The environment supplies a TeX group, so assignments divided into persistent and local forms acquire the documented lifetime without a separate restoration procedure.

Board column selection and board-option keys. A blank board column value means "use the persistent default". A non-blank value is normalized as a board-local override. The board key family provides local counterparts to the persistent setup keys and adds the valueless `break` switch. Start and reference dates use the same deferred strategy as persistent setup, so key order cannot alter ambiguous-date parsing.

```
\newcommand\kanban@setboardcolumns[1]{%
  \ifblank{#1}
    {\kanban@columnsboardinit}
    {\kanban@selectboardcolumns{#1}}}
\ExplSyntaxOn
\keys_define:nn {kanban/board}
  {
    break .code:n = {\booltrue{kanban@break}},
    break .value_forbidden:n = true,
    columns .code:n = {\kanban@setboardcolumns{#1}},
    date-format .code:n = {\kanban@selectboarddateformat{#1}},
    dateformat .meta:n = {date-format={#1}},
    startdate .code:n =
      {
        \tl_set:Nn \l__kanban_pending_startdate_tl {#1}
        \bool_set_true:N \l__kanban_pending_startdate_bool
      },
    reference-date .code:n =
      {
        \tl_set:Nn \l__kanban_pending_reference_date_tl {#1}
        \bool_set_true:N \l__kanban_pending_reference_date_bool
      },
    today .meta:n = {reference-date={#1}},
    palette .code:n = {\kanban@selectboardpalette{#1}},
    mode .code:n = {\kanban@selectboardmode{#1}},
    columnsep .code:n = {\kanban@setboardcolumnsep{#1}},
    labels .code:n = {\keys_set:nn {kanban/board-labels}{#1}},
    unknown .code:n =
```

```
      {\use:x {\msg_warning:nnnn {kanban-board}{unknown-key}
        {board}{\l_keys_key_str}}}
  }
```

A deferred board-option transaction. Board-option processing mirrors persistent setup but commits staged dates to the local snapshot setters. All immediate options, including a possible `date-format` override, are resolved before the dates are parsed. The transaction runs after the persistent values have been copied into the board, so omitted keys inherit their defaults.

```
\cs_new_protected:Npn \kanban@processboardoptions #1
  {
    \bool_set_false:N \l__kanban_pending_startdate_bool
    \bool_set_false:N \l__kanban_pending_reference_date_bool
    \keys_set:nn {kanban/board}{#1}
    \bool_if:NT \l__kanban_pending_startdate_bool
      {\exp_args:NV \kanban@setboardstartdate
        \l__kanban_pending_startdate_tl}
    \bool_if:NT \l__kanban_pending_reference_date_bool
      {\exp_args:NV \kanban@setboardreferencedate
        \l__kanban_pending_reference_date_tl}
  }
\ExplSyntaxOff
```

Per-column state and renderer-sequence construction. Each normalized column is represented internally by a numeric identifier. Setup stores the display name, initializes the board-qualified deferred task list and counter, assigns the cyclic palette background and foreground, and appends one adapter call to each renderer sequence. Card lists and counts carry the board number because those definitions are global during collection. Later columns in the break sequence are preceded by `\switchcolumn` so that `paracol` receives one stream per board column.

```
\newcommand\kanban@setupcolumn[2]{%
  \csdef{kanban@name@#2}{#1}%
  \csgdef{kanban@list@\kanban@boardgroup @#2}{}%
  \csgdef{kanban@count@\kanban@boardgroup @#2}{0}%
  \kanban@palettecolor{#2}\csedef{kanban@color@#2}{\kb@thiscolor}%
  \csedef{kanban@headingtext@#2}{\kb@thisheadingtext}%
  \numdef\kanban@ncols{\kanban@ncols+1}%
  \eappto\kanban@headseq{\noexpand\kanban@onehead{#2}}%
  \eappto\kanban@colseq{\noexpand\kanban@onecol{#2}}%
  \ifnumequal{#2}{1}
    {\eappto\kanban@colseq@b{\noexpand\kanban@onecol@b{#2}}}
    {\eappto\kanban@colseq@b{\noexpand\switchcolumn
       \noexpand\kanban@onecol@b{#2}}}}
```

Whole-board geometry validation. A syntactically valid non-negative gap may still be impossible for a board with many columns. Before rendering, the validator computes the total width of the $N - 1$ inter-column gaps and requires that total to be strictly smaller than `\linewidth`. An impossible value produces

an actionable error and falls back to zero for this board, which lets the document continue.

```
\newcommand\kanban@validateboardgeometry{%
  \ifnum\kanban@ncols>1\relax
    \kanban@totalgaps=\kanban@boardcolumnsep
    \multiply\kanban@totalgaps by \numexpr\kanban@ncols-1\relax
    \ifdim\kanban@totalgaps<\linewidth\relax\else
      \PackageError{kanban-board}
        {column separation leaves no width for columns}
        {For N columns, (N-1) times columnsep must be smaller than
          \string\linewidth.\MessageBreak
          The current board will use columnsep=0pt.}%
      \kanban@boardcolumnsep=0pt
    \fi
  \fi}
```

Environment entry and state snapshot. Opening a board either rejects unsupported nesting or creates a complete isolated snapshot before tasks are read. A rejected inner environment clears its render and active flags locally, sets the discard flag, and issues one error, so its body can be read safely without being attached to the outer board. The successful path resets the lifecycle flags, advances the unique board identifier, copies every persistent scalar and list default, processes local overrides, materializes the visual mode, clears the renderer sequences, registers all columns, validates the aggregate geometry, and suppresses source whitespace before the first task.

```
\newcommand\kanban@begin[1]{%
   \ifbool{kanban@active}
     {\boolfalse{kanban@render}%
      \boolfalse{kanban@active}%
      \booltrue{kanban@discardtasks}%
      \PackageError{kanban-board}
        {nested kanban environments are not supported}
        {Close the current board before opening another one.
          \MessageBreak The nested board was ignored.}}
     {\booltrue{kanban@render}%
   \boolfalse{kanban@break}%
   \boolfalse{kanban@discardtasks}%
   \booltrue{kanban@active}%
   \stepcounter{kanban@board}%
   \edef\kanban@boardgroup{\arabic{kanban@board}}%
   \edef\kanban@thestart{\kanban@startdate}%
   \edef\kanban@boarddateformat{\kanban@dateformat}%
   \edef\kanban@thereference{\kanban@referencedate}%
   \edef\kanban@boardmode{\kanban@mode}%
   \let\kanban@boardhighlabel\kanban@highlabel
   \let\kanban@boardmediumlabel\kanban@mediumlabel
   \let\kanban@boardlowlabel\kanban@lowlabel
   \let\kanban@boarddonelabel\kanban@donelabel
   \kanban@boardcolumnsep=\kanban@columnsep
   \kanban@paletteboardinit
   \kanban@columnsboardinit
   \kanban@processboardoptions{#1}%
   \kanban@applymode
   \def\kanban@ncols{0}\def\kanban@headseq{}%
   \def\kanban@colseq{}\def\kanban@colseq@b{}%
```

```
    \kanban@setupboardcolumns
    \kanban@validateboardgeometry
    \ignorespaces}}
```

\kanban
\endkanban

The public environment has one optional board-key list and no mandatory column argument. Columns come from the persistent default unless a board-local `columns` key overrides them. Closing the environment selects the page-breakable `paracol` renderer or the indivisible fixed raster. A rejected nested environment has its render flag cleared and therefore produces no board. Successfully rendered boards release their dynamically named state afterward, and the environment group restores all remaining local snapshots.

In break mode, the board gap is assigned to the standard `\columnsep` used by `paracol`, and the prepared sequence supplies headers, cards, and `\switchcolumn` calls. In fixed mode, a full-line `minipage` makes the board indivisible. The raster has one column per Kanban stage and two rows, all headers followed by all bodies, with equal heights within each row.

```
\NewDocumentEnvironment{kanban}{O{}}
  {\kanban@begin{#1}}
  {\ifbool{kanban@render}{\par\medskip
   \ifbool{kanban@break}
     {\setlength\columnsep{\kanban@boardcolumnsep}%
      \begin{paracol}{\kanban@ncols}\kanban@colseq@b\end{paracol}}
     {\noindent\begin{minipage}{\linewidth}%
      \begin{tcbraster}[
        raster columns=\kanban@ncols,
        raster rows=2,
        raster width=\linewidth,
        raster column skip=\kanban@boardcolumnsep,
        raster row skip=0pt,
        raster before skip=0pt,
        raster after skip=0pt,
        raster equal height=rows,
        raster valign=top]
        \kanban@headseq\kanban@colseq
      \end{tcbraster}%
      \end{minipage}}%
   \par\vspace{2mm}\kanban@cleanupboard}{}}
⟨/package⟩
```

Change History

v1.0.0

General: Initial public release. 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

__kanban_canonical_date: 645, 869, 881, 893, 905
__kanban_classify_calendar_date: . . . 758, 819
__kanban_classify_project_month: . . . 778, 836
__kanban_column_normalize:nN . . . 346, 361, 419
__kanban_columns_normalize:nNN . 355, 380, 405
__kanban_columns_select_global:n . . . 378, 393
__kanban_deadline_diagnose:n 842, 861
__kanban_dimension_token:N 950
__kanban_dimension_token:NTF 978
__kanban_dimension_valid:n 966
__kanban_dimension_valid:nTF 987
__kanban_four_digits:n 638, 647, 662
__kanban_load_reference_date: . . 737, 760, 780
__kanban_palette_define:nn 173, 336
__kanban_palette_define_pairs:nn . . . 225, 335
__kanban_palette_pair:n 199, 241
__kanban_palette_select_board:n . . . 283, 345
__kanban_palette_select_global:n . . . 264, 343
__kanban_parse_date:nn 569, 654, 817, 826, 844, 867, 879, 891, 903
__kanban_parse_project_month:n . 716, 823, 850
__kanban_set_components:nnn 561, 581, 587, 593, 606, 607
__kanban_set_match:Nn 556, 565, 566, 567
__kanban_set_max_day: 511, 544, 790
__kanban_two_digits:n . . 633, 648, 649, 658, 660
__kanban_validate_date: 535, 619, 630
__kanban_year_divisible:n 504
__kanban_year_divisible_p:n 519, 521, 522

B

\bool_if:NF 821, 845
\bool_if:NT 618, 655, 824, 828, 1127, 1130, 1460, 1463
\bool_if:nT 620, 851
\bool_if:NTF 242, 818, 847, 868, 880, 892, 904, 1277
\bool_if:nTF 210, 517, 601
\bool_set_false:N . . 231, 537, 571, 572, 573, 574, 718, 719, 1124, 1125, 1259, 1273, 1457, 1458
\bool_set_true:N . . . 215, 223, 550, 563, 564, 600, 613, 725, 733, 1107, 1112, 1267, 1439, 1444

C

\c__kanban_tex_lengths_str 910, 958
\clist_clear:N 229, 230, 296, 357
\clist_count:N 311
\clist_gclear:N 194, 276
\clist_gset:NV 271, 274
\clist_gset_eq:NN 192, 255, 257, 386
\clist_if_empty:NF 408
\clist_if_empty:NTF 181, 236, 320, 383
\clist_item:Nn 316, 325
\clist_map_function:NN 240
\clist_map_inline:Nn 438
\clist_map_inline:nn 359
\clist_put_right:NV 217, 219, 371
\clist_set:Nn 180, 235
\clist_set:NV 290, 294
\clist_set_eq:NN 303, 305, 396, 410
\clist_use:Nn 434

E

\endkanban 1527
environments:
kanban *2*
\exp_args:NNx 958
\exp_not:V 183, 238, 244, 281, 299, 457

G

\g__kanban_column_display_prop . . . 88, 388, 399
\g__kanban_columns_clist 87, 386, 397
\g__kanban_package_options_tl 1140, 1143, 1144, 1146, 1150, 1151, 1152, 1156, 1161
\g__kanban_palette_clist . 70, 192, 255, 271, 304
\g__kanban_palette_fg_clist 71, 194, 257, 274, 276, 306
\g__kanban_palette_name_tl . . 72, 190, 253, 277
\g__kanban_palette_texts_prop 69, 187, 249, 272, 292
\g__kanban_palettes_prop . 68, 185, 246, 268, 287
\group_begin: 1158
\group_end: 1162

I

\int_case:nnF 513
\int_compare:nNnT 538, 540, 542, 545, 547, 635, 640, 641, 642, 773, 793, 798, 807
\int_compare:nNnTF 202, 506, 769, 802
\int_decr:N 801
\int_incr:N 440
\int_mod:nn 313, 507
\int_set:Nn 310, 312, 525, 526, 528, 529, 530, 531, 533, 558, 731, 742, 743, 744, 750, 752, 754, 761, 765, 781
\int_set_eq:NN 626, 627, 629, 786, 788, 791, 796, 830, 832, 834

\int_step_inline:nn 461
\int_to_arabic:n 636, 643
\int_use:N 442
\int_zero:N 437
\iow_char:N 115, 124, 166

K

\kanban 1527
kanban (env.) *2*
\kanban@applymode 1055, 1521
\kanban@applypackageoptions 1154, 1197
\kanban@authorback 51, 1211
\kanban@authorchip 1228, 1322
\kanban@authortext 52, 1212
\kanban@availablecolumns 431, 1384, 1391
\kanban@begin 1494, 1528
\kanban@boardcolumnsep 35, 1051, 1483, 1491, 1517, 1531, 1538
\kanban@boarddateformat 44, 654, 664, 817, 827, 844, 879, 903, 1034, 1510
\kanban@boarddonelabel 66, 1250, 1516
\kanban@boardgroup 33, 467, 469, 1303, 1340, 1344, 1350, 1352, 1397, 1399, 1401, 1470, 1471, 1508
\kanban@boardhighlabel 63, 1217, 1513
\kanban@boardlowlabel 65, 1223, 1515
\kanban@boardmediumlabel . . 64, 1219, 1221, 1514
\kanban@boardmode . . . 46, 1008, 1009, 1056, 1512
\kanban@bordercolor 50, 1060, 1066, 1306
\kanban@candidatecolumnsep 36, 1037, 1038, 1041, 1047, 1048, 1051
\kanban@canonicaldateformat . . . 1011, 1026, 1031
\kanban@cardcolor . . . 48, 1058, 1064, 1313, 1315
\kanban@cardtint 49, 1059, 1065, 1315
\kanban@cleanupboard 459, 1547
\kanban@collectkeyvalue 1148, 1176, 1178, 1180, 1182, 1184, 1186, 1188, 1190, 1192, 1194
\kanban@collectpackageoption . 1141, 1168, 1174
\kanban@colseq 1476, 1523, 1544
\kanban@colseq@b 1478, 1479, 1523, 1532
\kanban@columnsboardinit 394, 1426, 1519
\kanban@columnsep 34, 38, 1041, 1517
\kanban@completedchip 1247, 1322
\kanban@dateformat 43, 867, 891, 1029, 1510
\kanban@deadlineback 53, 1213
\kanban@deadlinechip 1234, 1324
\kanban@deadlinetext 54, 1214
\kanban@diagnosedeadline 860, 1411
\kanban@dimensionwarning 988, 1039, 1044, 1049, 1054
\kanban@donelabel 62, 1516
\kanban@dotask 1375, 1420
\kanban@headingtext 55, 321, 1061, 1067
\kanban@headseq 1475, 1522, 1544
\kanban@highlabel 59, 1513
\kanban@hightext 56, 1218, 1238
\kanban@lowlabel 61, 1515
\kanban@lowtext 58, 1224, 1249
\kanban@mediumlabel 60, 1514
\kanban@mediumtext 57, 1220, 1222, 1242
\kanban@mode 45, 1003, 1004, 1512
\kanban@ncols 461, 1474, 1482, 1484, 1522, 1532, 1535
\kanban@normalizechoice 451, 1002, 1007, 1012, 1289
\kanban@normtaskcol 417, 1380
\kanban@onecol 1342, 1476
\kanban@onecol@b 1346, 1478, 1480
\kanban@onehead 1336, 1475
\kanban@paletteboardinit 301, 1518
\kanban@palettecolor 308, 1472
\kanban@preparedeadline 651, 1236
\kanban@prioritypill 1215, 1320
\kanban@processboardoptions 1455, 1520
\kanban@processsetup 1122, 1135, 1160
\kanban@processtaskoptions 1372, 1378
\kanban@referencedate 41, 889, 893, 1511
\kanban@rendercard 1316, 1403
\kanban@resolvecardback 1311, 1317
\kanban@selectboardcolumns 401, 1427
\kanban@selectboarddateformat 1030, 1434
\kanban@selectboardmode 1006, 1448
\kanban@selectboardpalette 344, 1447
\kanban@selectcolumns 392, 1000, 1100
\kanban@selectdateformat 1025, 1102
\kanban@selectmode 1001, 1115
\kanban@selectpalette 342, 999, 1101
\kanban@setbackground 1252, 1366
\kanban@setboardcolumns 1424, 1433
\kanban@setboardcolumnsep 1045, 1449
\kanban@setboardlabel 1073, 1090, 1091, 1092, 1093
\kanban@setboardreferencedate 898, 1464
\kanban@setboardstartdate 874, 1461
\kanban@setcolumnsep 1035, 1116
\kanban@setlabel . . . 1069, 1080, 1081, 1082, 1083
\kanban@setpriority 1288, 1358
\kanban@setreferencedate 886, 1131
\kanban@setstartdate 862, 1128
\kanban@setupboardcolumns 429, 1524
\kanban@setupcolumn 447, 1468
\kanban@startdate 39, 865, 869, 1509
\kanban@textcolor 47, 1057, 1063, 1307
\kanban@thereference . . . 42, 739, 901, 905, 1511
\kanban@thestart 40, 827, 877, 881, 1509
\kanban@totalgaps 37, 1483, 1484, 1485
\kanban@validateboardgeometry 1481, 1525
\kanban@validatedimensionTF . . . 986, 1036, 1046
\kanban@warning 169, 1005, 1010, 1028, 1033, 1071, 1075, 1138, 1423
\kanbandefinepalette *14*, 332, 993, 996, 998
\kanbansetup *5*, 1135
\kanbantask *6*, 1386, 1419
\kb@cardback 1313, 1315, 1318
\kb@choice 457, 1003, 1004, 1008, 1009, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1291, 1292, 1293, 1294, 1295
\kb@clock 1230, 1240, 1244, 1246
\kb@datechoice 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1027, 1029, 1032, 1034
\kb@displaydeadline 653, 669, 678, 687, 696, 705, 1240, 1244, 1246
\kb@evaloverdue 813, 1235
\kb@pc 1216, 1218, 1220, 1222, 1224, 1226
\kb@pl . . 1216, 1217, 1219, 1221, 1223, 1225, 1227
\kb@pt 1216, 1218, 1220, 1222, 1224, 1226
\kb@thiscolor 319, 1472
\kb@thisheadingtext 321, 328, 1473
\kbdefinepalette *14*, 332
\kbset 1135
\kbtask *6*, 1419
\kbtask@author 1359, 1376, 1406
\kbtask@bg 1256, 1278, 1377, 1410
\kbtask@column 420, 1381
\kbtask@columnid 421, 425, 1388, 1398, 1400, 1402, 1404
\kbtask@completed 1364, 1377, 1409

\kbtask@deadline 1360, 1376, 1407, 1412
\kbtask@late 1362, 1377, 1408
\kbtask@prio 1290, 1291, 1292, 1293, 1294, 1376, 1405
\keys_define:nn 1078, 1088, 1098, 1356, 1429
\keys_set:nn 1117, 1126, 1373, 1450, 1459

L

\l__kanban_ambiguous_slash_bool 502, 574, 600, 622
\l__kanban_background_ok_bool 102, 1259, 1267, 1273, 1277
\l__kanban_background_seq 101, 1269, 1270
\l__kanban_background_tl 100, 1254, 1255, 1257, 1265, 1269, 1278, 1283
\l__kanban_bad_palette_bool 86, 215, 223, 231, 242
\l__kanban_board_column_display_prop 90, 398, 412, 445
\l__kanban_board_columns_clist 89, 396, 410, 434, 438
\l__kanban_board_palette_clist 73, 290, 303, 311, 316
\l__kanban_board_palette_fg_clist 74, 294, 296, 305, 320, 325
\l__kanban_choice_tl 99, 453, 454, 455, 456, 457
\l__kanban_column_display_tl 97, 348, 349, 351, 373, 446, 448
\l__kanban_column_display_tmp_prop 92, 382, 389, 407, 413
\l__kanban_column_id_tl 98, 424, 426, 441, 444, 448
\l__kanban_column_ids_prop . . 93, 423, 436, 443
\l__kanban_column_index_int . . 94, 437, 440, 442
\l__kanban_column_tl 96, 361, 362, 364, 367, 371, 372, 419, 420, 423
\l__kanban_columns_tmp_clist 91, 381, 383, 387, 406, 408, 411
\l__kanban_current_month_int 484, 781, 801, 803, 808
\l__kanban_date_components_bool 501, 564, 573, 618
\l__kanban_date_syntax_bool 500, 563, 572, 613, 821, 847
\l__kanban_deadline_day_int 476, 545, 548, 567, 626, 627, 649, 658, 764, 831
\l__kanban_deadline_key_int . 481, 761, 770, 774
\l__kanban_deadline_match_seq 475, 559, 580, 586, 592, 598, 729, 732, 749, 751, 753, 755
\l__kanban_deadline_max_day_int 479, 525, 526, 528, 529, 530, 531, 533, 549, 795, 797
\l__kanban_deadline_month_int 477, 513, 540, 542, 566, 628, 629, 648, 660, 763, 788, 833
\l__kanban_deadline_tl 474, 575, 576, 579, 585, 591, 597, 612, 720, 721, 724, 728
\l__kanban_deadline_year_int 478, 507, 538, 565, 647, 662, 762, 786, 835
\l__kanban_dimension_tl 497, 968, 969, 973, 976, 979
\l__kanban_display_day_tl 493, 657, 672, 680, 689, 700, 709
\l__kanban_display_format_tl 496, 663, 665
\l__kanban_display_month_tl 494, 659, 671, 681, 690, 699, 708
\l__kanban_display_year_tl 495, 661, 673, 682, 691, 698, 707
\l__kanban_pair_bg_tl 80, 204, 208, 212, 218
\l__kanban_pair_fg_tl 81, 206, 209, 213, 220
\l__kanban_palette_bg_clist 76, 217, 229, 248, 256
\l__kanban_palette_bg_tl . 78, 269, 271, 288, 291
\l__kanban_palette_colors_clist 75, 180, 181, 186, 193, 235, 236, 240
\l__kanban_palette_count_int 83, 310, 313
\l__kanban_palette_fg_clist 77, 219, 230, 251, 258
\l__kanban_palette_fg_tl . 79, 273, 275, 293, 295
\l__kanban_palette_index_int . 84, 312, 317, 326
\l__kanban_palette_item_tl 82, 314, 319, 323, 329
\l__kanban_palette_name_tl 95, 175, 176, 177, 183, 186, 188, 189, 227, 228, 232, 238, 244, 247, 250, 252, 266, 267, 269, 273, 278, 281, 285, 286, 288, 293, 299
\l__kanban_palette_pair_seq 85, 201, 202, 205, 207
\l__kanban_pending_reference_date_bool . . 106, 1112, 1125, 1130, 1444, 1458, 1463
\l__kanban_pending_reference_date_tl 104, 1111, 1132, 1443, 1465
\l__kanban_pending_startdate_bool 105, 1107, 1124, 1127, 1439, 1457, 1460
\l__kanban_pending_startdate_tl 103, 1106, 1129, 1438, 1462
\l__kanban_project_boundary_day_int 488, 791, 794, 796, 800
\l__kanban_project_month_int 483, 731, 804, 809
\l__kanban_project_month_syntax_bool 503, 719, 725, 853
\l__kanban_project_start_day_int 485, 792, 830
\l__kanban_project_start_month_int 486, 785, 832
\l__kanban_project_start_year_int 487, 783, 834
\l__kanban_reference_day_int 489, 744, 754, 768, 799
\l__kanban_reference_key_int 482, 765, 770, 774
\l__kanban_reference_month_int 490, 743, 752, 767, 784, 789
\l__kanban_reference_tl 492, 739, 740, 749
\l__kanban_reference_year_int 491, 742, 750, 766, 782, 787
\l__kanban_swap_int 480, 626, 629
\l__kanban_valid_date_bool 498, 537, 550, 571, 623, 655, 818, 828, 845, 868, 880, 892, 904
\l__kanban_valid_project_month_bool 499, 718, 733, 824, 854
\l_keys_key_str 1086, 1096, 1120, 1370, 1453

M

\msg_line_context: . 108, 112, 117, 120, 123, 128, 131, 136, 141, 144, 148, 153, 157, 162, 1159
\msg_warning:nn 178, 233, 384
\msg_warning:nnn 243, 848, 856, 1282, 1296
\msg_warning:nnnn 170, 182, 237, 280, 298, 339, 870, 882, 894, 906, 989, 1085, 1095, 1119, 1369, 1452

P

\prg_new_conditional:Npnn 504
\prop_clear:N 358, 436
\prop_get:NnN 445
\prop_get:NVNTF 268, 272, 287, 292, 422
\prop_gput:NVV 185, 246, 249
\prop_gremove:NV 187
\prop_gset_eq:NN 388
\prop_if_in:NVTF 364
\prop_put:NnV 443
\prop_put:NVV 372
\prop_set_eq:NN 398, 412

R

\regex_extract_all:nnN 1268
\regex_extract_once:nnN 747
\regex_extract_once:nnNTF 577, 583, 589, 595, 726

\regex_match:nnT 610, 722, 1260
\regex_match:nnTF 970

S

\seq_count:N 202
\seq_item:Nn . . . 205, 207, 559, 732, 751, 753, 755
\seq_map_inline:Nn 1270
\seq_set_split:Nnn 201
\str_case:VnF 665
\str_const:Nx 910
\str_if_eq_p:ee 603, 604
\str_if_in:NnTF 958

T

\text_expand:n 433, 575, 720, 968, 1254
\text_lowercase:n 456
\text_purify:n 351
\text_uppercase:n 352
\tl_gput_right:Nn 1144, 1145, 1151, 1152
\tl_gset_eq:NN 277
\tl_if_blank:nF 403, 861
\tl_if_blank:nTF 864, 876, 888, 900
\tl_if_blank:VF 362, 1257
\tl_if_blank:VTF 177, 232, 740
\tl_if_blank_p:V 212, 213
\tl_if_empty:NF 1143, 1150, 1156
\tl_if_eq:NNT 189, 252
\tl_if_single_token:nTF 976
\tl_set:Nn 175, 227, 266, 285, 348, 453, 1106, 1111, 1438, 1443
\tl_set:Nx . 204, 206, 314, 323, 350, 352, 441, 455, 575, 657, 659, 661, 663, 720, 739, 968, 1254
\tl_to_str:N 1283
\tl_trim_spaces:N 176, 208, 209, 228, 267, 286, 349, 353, 454, 576, 721, 969, 1255
\tl_use:N 319, 329, 367, 420, 426, 1278
\token_if_dim_register:NTF 952
\token_if_skip_register:NTF 955
\token_to_meaning:N 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 959

U

\use:x 182, 237, 243, 280, 298, 1085, 1095, 1119, 1280, 1369, 1452