Contents

Build a simple sales-document package **2**
1. Create the directory . . . 3
Identity, versions, replacement, and recovery . . . 3
2. Add the template . . . 3
3. Add styles . . . 4
4. Create the import ZIP and verify . . . 4

Build a timesheet package **5**
Kind-specific presentation . . . 6
Coordinated page design . . . 7
Import and responsibility . . . 7

Build a simple sales-document package

This walkthrough builds a deliberately simple package. It hard-codes most visual choices and uses only the essential public contract: the manifest, page margins, prepared document items, and the application-owned financial summary. **Experimental:** Template API v1 is not stable; retest this package with each application release.

Validating, importing, or repairing a custom package requires the **Pro** or **Ultimate** tier. The package format and authoring contract themselves are independent of licensing.

Custom-package import is available from the layout-template editor for both sales-document and timesheet layouts. The complete sales-document example is maintained under `examples/simple/` and is compiled and rendered by the repository contract tests.

The first rendered page looks like this with the maintained English fixture:

TechVision Solutions GmbH

Order Confirmation

Document number: OC-BASE-001 · Document date: Aug 19, 2026

Example Engineering Ltd.
Customer Street
54321 Customer City
DE

Position	Description	Quantity	Total
1	Implementation and commissioning 1 – deterministic long description for pagination and wrapping verification	1 hour	€76.25
2	Implementation and commissioning 2 – deterministic long description for pagination and wrapping verification	1 hour	€77.50
3	Implementation and commissioning 3 – deterministic long description for pagination and wrapping verification	1 hour	€78.75
4	Implementation and commissioning 4 – deterministic long description for pagination and wrapping verification	1 hour	€80.00
5	Implementation and commissioning 5 – deterministic long description for pagination and wrapping verification	1 hour	€81.25
6	Implementation and commissioning 6 – deterministic long description for pagination and wrapping verification	1 hour	€82.50
7	Implementation and commissioning 7 – deterministic long description for pagination and wrapping verification	1 hour	€83.75

The fixture deliberately contains enough items to demonstrate pagination; the financial summary continues on the second page.

1. Create the directory

Create these three files:

```
manifest.json
document.peb
styles.css
```

Copy the example manifest. It declares one `sales_document` entry point, all currently supported document families, Template API v1, and only `page.margins` as an honored setting. See the generated reference for manifest compatibility and diagnostics.

Identity, versions, replacement, and recovery

Every manifest requires a string `templateVersion`, for example `"1"` or `"2026-autumn"`. It is independent of `packageFormatVersion` and `templateApiVersion`; versions are not sorted or selected automatically. The maintained examples start at `"1"`.

A supplied `packageId` determines identity. If omitted, identity is the uploaded ZIP filename without its final `.zip` extension: `invoice.zip` with version `"2"` installs under `template-packages/invoice/version-2/`. Renaming that ZIP changes identity; use a stable `packageId` to avoid this. An invalid supplied ID is rejected rather than replaced with the filename.

Identity and version must be portable, nonempty NFC names of at most 100 UTF-8 bytes. Path separators, traversal, platform-reserved names, and case-equivalent collisions are rejected. Names remain literal directory components. The application performs the complete filesystem and resource validation; the JSON schema alone does not establish importability.

Uploading the same identity/version replaces its complete contents. Removed resources do not remain installed, and failed replacement preserves the previous usable package. Choose a different version when existing references must continue to use the old layout. **Replacing a version changes later renders of all documents using it, including published sales documents and frozen timesheet periods.** Their references remain unchanged; already stored or exported PDF files are not rewritten.

Connected Client previews fetch the current package from the Host on every resolution. During a connectivity failure, a valid last download from the same dataset may be used; it can be stale. The next resolution after reconnect refreshes it. Missing Host content, permission, dataset, or validation errors do not permit stale fallback. Final Client exports use Host content.

If a preview reports missing or unusable content, upload the displayed identity/version again through the normal template editor. A different version cannot restore that reference. Re-upload requires the usual Pro or Ultimate license and writer permission. There is no startup scan or substitution approval. Keep source ZIPs or a separate filesystem backup: database backups do not include package files. Registered production and sandbox datasets share the installed package contents.

2. Add the template

Use this complete `document.peb`:

```
<!DOCTYPE html>
<html lang="{{ common.locale }}">
<head>
  <meta charset="UTF-8" />
  <style>{{ common.resources.fontCss | raw }}</style>
  <style>{% include "styles.css" %}</style>
  <style>
    @page {
      size: {{ settings.page.size }};
      margin: {{ settings.page.marginTopMm }}mm {{ settings.page.marginRightMm }}mm {{ settings.page.marginBottomMm }}mm
      {{ settings.page.marginLeftMm }}mm;
    }
  </style>
```

```
</head>
<body>
  <div>
    <p class="eyebrow">{{ common.tenant.displayName }}</p>
    <h1>{{ document.title }}</h1>
    <p>{{ common.labels.documentNumber }}: {{ document.number }} · {{ common.labels.documentDate }}: {{ document.date
     }}</p>
  </div>

  <div class="recipient">
    <strong>{{ common.recipient.displayName }}</strong>
    {% for line in common.recipient.addressLines %}<div>{{ line }}</div>{% endfor %}
  </div>

  <table class="items">
    <thead><tr><th>{{ common.labels.position }}</th><th>{{ common.labels.description }}</th><th>{{ common.labels.
     quantity }}</th><th>{{ common.labels.total }}</th></tr></thead>
    <tbody>
    {% for item in document.items %}
      <tr><td>{{ item.displayPosition }}</td><td>{{ item.descriptionXhtml | raw }}</td><td>{{ item.quantityWithUnit }}</
     td><td>{{ item.lineTotal }}</td></tr>
    {% endfor %}
    </tbody>
  </table>

  {% if components.financialSummary.available %}{% include "invoicer/components/v1/financial-summary.peb" %}{% endif %}
</body>
</html>
```

`common.resources.fontCss` is application-prepared safe CSS for the maintained embedded fonts and any validated package-local faces. Include it before package styles. `descriptionXhtml` is application-prepared safe XHTML, so these explicit fields are rendered with the only supported filter, `raw`. Money, VAT grouping, totals, labels, dates, and quantities are already prepared display values. Do not calculate them in Pebble.

The reserved financial-summary include reads `components.financialSummary`; it accepts no arguments. Its stable outer hook and semantic row hooks are available for CSS. Target a role with `[data-row-role="vat_bucket_tax"]` or `.inv-c-role-vat_bucket_tax`; do not infer row meaning from `data-row-id`. See application-owned components.

The prepared financial-summary rows are authoritative PDF presentation. When `document.showVatRateColumn` is `false` for an eligible homogeneous zero-tax rule, the application omits the redundant net and zero-VAT rows and supplies a neutral `Total` row; rebate calculation inputs remain available. This does not change structured e-invoice totals or tax data.

3. Add styles

Copy `styles.css`. The example applies the declared page margins and styles the component via `[data-invoicer-component="financial-summary"]`.

4. Create the import ZIP and verify

Create a ZIP from inside the source directory so `manifest.json`, `document.peb`, and `styles.css` are at the archive root. Do not ZIP the containing directory. Review the identity/version and replacement notice before confirming import.

ZIPs created by macOS Finder are accepted. The importer ignores zero-byte directory records and known Finder metadata (`__MACOSX`, `.DS_Store`, and `._` files) after validating their bounded paths. A directory record such as `images/` is only container metadata: declared files below it, such as `images/brand.png`, remain part of the package and are validated normally.

ZIP the contents of the package root so `manifest.json` remains at the ZIP root, then import it from the layout-template editor. During development, strict rendering reports missing variables as `PRESENTATION_INCOMPATIBLE`, missing files as `RESOURCE_MISSING`, and unsupported source constructs as `PEBBLE_FEATURE_FORBIDDEN`. The complete list is in diagnostics.

For package-local branding, declare each PNG or JPEG under `resources.images`, include the exact file in the ZIP, and reference its normalized package-relative path from XHTML or CSS, for example `<img src="images/brand.png" alt="" />`

or `background-image: url('images/brand.png')`. Inline `data:` resources and filesystem or network URLs are not authoring interfaces.

For a package-local font, declare every static TrueType face explicitly:

```
"fonts": [
  {"path": "fonts/Acme-Regular.ttf", "family": "Acme", "weight": 400, "style": "normal"},
  {"path": "fonts/Acme-Bold.ttf", "family": "Acme", "weight": 700, "style": "normal"},
  {"path": "fonts/Acme-Italic.ttf", "family": "Acme", "weight": 400, "style": "italic"}
]
```

Then select `font-family: 'Acme', 'IBM Plex Sans', sans-serif` and the declared weight/style from package CSS. The family, weight, and style must match the font's internal metadata. A package may contain at most eight faces, each at most 1 MiB and together at most 4 MiB. OTF, TTC, WOFF/WOFF2, variable, color, bitmap, Type 1, operating-system, local-file, and network fonts are unsupported. The importer must have the right to embed and distribute the supplied font software. The professional starters provide complete sales-document and timesheet examples; this simple example deliberately leaves `fonts` empty to demonstrate the maintained IBM Plex fallback.

The repository verifies this exact source against English and German fixtures. Rendering success demonstrates contract compatibility, not legal or visual approval of a customer layout.

When you are ready to begin from a fuller design, use the professional sales-document starter. Its localized page counter sits in the upper-right running header at the original header height, left-aligned with the text in the repeating right-side information column. The deliberately small example in this walkthrough remains useful for learning the minimum contract, but it is not the finished starter package.

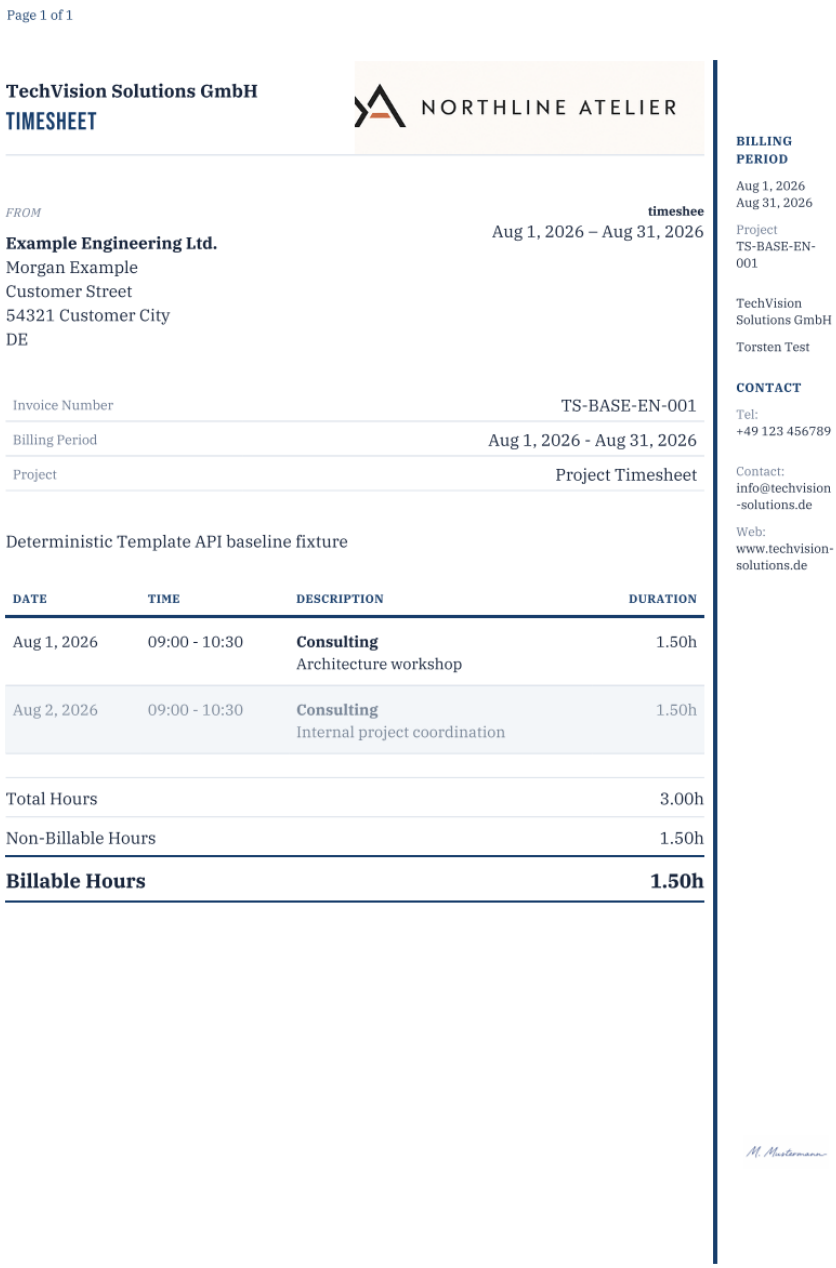
Build a timesheet package

A timesheet package uses the same ZIP, manifest, resource, validation, author-version, and Client/Host rules as a sales-document package. Its manifest declares `templateKind: timesheet`, an empty `supportedDocumentFamilies` list, and a timesheet entry point such as `timesheet.peb`.

Validating, importing, or repairing a custom package requires the **Pro** or **Ultimate** tier. An already installed custom timesheet package may remain in use below Pro, but importing or repairing one still requires Pro. This product rule does not change the Template API contract.

The complete adoptable source is the professional timesheet starter. ZIP the contents of that directory so `manifest.json` is at the archive root, or use the ready-to-import ZIP.

With the maintained English single-page fixture, the rendered starter looks like this:



Kind-specific presentation

Exactly one kind root is populated during rendering. A timesheet template reads `timesheet`; `document` is null. Use the prepared fields rather than rebuilding period, duration, localization, or billable-summary rules:

- `timesheet.documentTitle`, `number`, `periodStart`, and `periodEnd` identify the frozen period;
- `timesheet.metadata` contains prepared labeled metadata;
- `timesheet.entries` contains prepared dates, optional time ranges, safe description XHTML, optional categories and locations, display durations, and billable state;
- `timesheet.hasLocations` determines whether the location column is useful;
- `invoicer/components/v1/timesheet-summary.peb` renders the application-owned prepared totals.

Null checks are part of the authoring contract. A time range, category, location, description, reference, logo, or footer

label may be absent. Lists remain present and may be empty.

Coordinated page design

The starter shares its package-local IBM Plex Serif regular, bold, and italic faces, its title-only Bebas Neue regular face, colors, tenant-logo treatment, and repeating right-side information column with the sales-document starter. It puts the prepared period and project information above the vertical company-information rows and prints the page counter in the running header. It deliberately emits no conventional bottom footer.

The package-local signature mark is a fictional visual sample reading `M. Mustermann`, the conventional German placeholder name. Replace or remove it before customer use. The tenant logo is different: it comes from `common.resources.tenantLogo`, is supplied by the application, and may be null.

Import and responsibility

Review the package locally, inspect the accepted font faces, confirm that the importing organization may embed and distribute the supplied font software, and import it through a timesheet layout-template editor. In Client mode the Host repeats validation and installation, while connected local previews refresh validated package contents from the Host into the dataset-scoped cache. Frozen periods retain their identity/version reference; replacing that version changes later renders. See the version and recovery rules.

Successful import and rendering prove contract compatibility only. The importing company remains responsible for the resulting timesheet's visual, textual, and legal suitability.