Mail Templates: subject, body and whitelabeling

A mail needs something to say. **Mail Templates** are the subject and the body, written once and reused by every render that mails a document. They are edited in a WYSIWYG editor, so nobody has to write HTML, and everything variable is a placeholder that VeloxFactory fills in at send time: the trace id, a parameter of the request, the name of the report, a column of the data that was rendered.

A template decides the content. The surrounding mail, a plain card in your application's colours, is built by VeloxFactory, and a template can override parts of it when a customer needs their own look.

Screenshot placeholder: `mail-template.edit.png`
Mail Template edit page with the rich text editor, the toolbar and the placeholder and preview buttons.

Where to find it

Mail Templates live in the main menu under **Configuration → Mail Templates**. The overview is a regular VeloxFactory lookup with search over name, subject and description, a filter for the active state and the usual audit filters.

A template has a unique name, which is what a render request may use instead of the ID, a subject, a body and an active flag. An inactive template stays in the master data but is no longer offered and no longer resolves.

The editor

The body is written in a rich text editor built into VeloxFactory. No add-on, no external service, and nothing to learn: the toolbar does what a toolbar does.

Group	Tools
Text	Bold, italic, underline, headings, paragraph, bullet and numbered lists, text colour and remove colour, clear formatting.
Links	Insert, edit and remove links. With the caret inside an existing link, the link tool edits that link instead of creating a new one. With nothing selected, it inserts the URL itself as the link text. Only `http`, `https` and `mailto` are accepted, and every link is marked so it opens safely.
Tables	Insert a table with a chosen number of rows and columns and an optional header row, then insert or delete rows and columns from wherever the caret sits.
Images	Insert a graphic from the Common Report Resources, with an optional width.
Buttons	Turn paragraphs into a button, change its colours or remove it again. See **Buttons and callouts** below.
Callouts	Frame paragraphs as an Info, Warning or Danger box, change its type or remove it again. See **Buttons and callouts** below.
Placeholders	Insert a placeholder at the caret, chosen from the catalog.

Pasting is deliberately plain text. Formatted content dragged in from Word or a browser is the single most common reason a mail looks broken in one client and fine in another, so the editor strips it and keeps the words.

Whatever the editor produces is cleaned on the server against an allowlist before it is stored. That is not a formality: it is what guarantees that a template cannot carry a script, a tracking pixel or an arbitrary external image, no matter what was pasted into it.

Placeholders

Anything in square brackets is replaced when the mail is built. Both the subject and the body support them.

> Hello [contactName], attached is your report [report.name] with the trace ID [traceId], rendered on [now.dateLocale].

The **Placeholders** button opens a dialog listing every token that is available, grouped, with a description and an example, and each one copyable with a click. Opened from a template that is tied to a report, the parameter and field tokens are expanded to the real names of that report.

Group	Examples
General	`[traceId]`, `[reportFileName]` (the attachment name), `[reportUrl]`, `[outputType]`, `[appName]`, `[contactName]`, `[broadcastId]`
Generic	`[now.date]`, `[now.dateLocale]`, `[now.dateTime]`, `[now.weekday]`, `[now.monthName]`, `[now.weekNumber]`, `[now.quarter]`
Report	`[report.name]`, `[report.description]`, `[report.context]`, `[report.connection]`
History Record	`[historyRecord.id]`, `[historyRecord.status]`, `[historyRecord.url]` as a direct link into VeloxFactory
User, API Token	`[user.name]`, `[user.email]`, `[token.name]` - who or what triggered the render
Printer	`[printer.name]`, `[printer.copies]` when a print task was created alongside
Parameters, Data	`[parameters.P_ORDER_NO]`, `[data.count]`, `[data.first.CUSTOMER]`, `[data.last.ARTICLE]`
Recipients	`[recipients.toCount]`, `[recipients.ccCount]`, `[recipients.bccCount]`
Environment	`[env.appEnv]`, `[env.hostname]`, `[env.timezone]` - to mark a mail from a test system as such
Images	`[image.logo_dark]` - one entry per usable Common Report Resource

> **An unknown placeholder never breaks a mail.** It resolves to an empty string and is reported back to the caller, so a typo shows up as a gap and a note, not as a failed send.

> **Placeholder values are inserted as text.** A resolved value that happens to contain markup, for example a customer name with a `<b>` in it, appears literally in the mail instead of changing its formatting. The markup you set in the editor is unaffected.

The same catalog is available beyond subject and body: recipients, contact name and the names the attachments carry in the mail accept placeholders too. See the Report Mailing page for how that behaves.

Graphics

Mail graphics are the **Common Report Resources** you already maintain, the same PNG and JPEG files the reports use. A logo is uploaded once and serves both a printed label and a mail.

Insert one with the image button, or type its placeholder, for example `[image.logo_dark]`. Both produce the same thing, and the editor shows the graphic while you write.

At send time the graphic is embedded **into** the mail as an inline attachment, not linked from a server. That matters for two reasons: the mail is self-contained, so it also works for a recipient outside your network, and no client shows a *load external images* bar or treats it as a tracking pixel.

> ℹ **Only graphics from the master data.** An image pasted in from a website is removed when the template is saved. If a graphic should be in a mail, upload it under Configuration → Common Report Resources and reference it by name.

A Common Report Resource that a template embeds cannot be deleted or renamed while the template uses it. The refusal names the template, so it is clear what would break.

Buttons and callouts

A report mail usually has one thing the recipient should do, open the report, confirm an order, check a status, and sometimes one thing they must not overlook. **Buttons** and **callouts** make both stand out, without anyone writing HTML and without fighting the rendering quirks of Outlook.

Screenshot placeholder: `mail-template.blocks.png`
The editor with a button, its colour popover and the three callout types.

Buttons

The button tool in the toolbar (icon `smart_button`) turns the paragraph the caret is in into a button. With a selection, every selected paragraph becomes a button. On an empty line it inserts a new

button with the text *Button text*, ready to be overwritten.

A button holds whatever a paragraph holds. Links are set with the usual link tool, and the URL can mix text and placeholders, for example `https://example.com/report/[traceId]`. If the button consists of nothing but a link, the whole button area is clickable, not only the words. Centre the paragraph inside the button and the button itself is centred in the mail.

Setting	Behaviour
Background	The header colour of the mail by default. That is the colour the mail actually uses, so a template with its own header colour override gets buttons in that colour. The checkbox **Header color** returns a changed background to this default.
Text	White by default. The checkbox **White** returns a changed text colour to this default.

Both colours are set per button in the popover of the button tool. To change an existing button, click into it and then on the tool: **Apply** stores the new colours, and the same popover removes the button again.

Callouts

The callout tool (icon `format_quote`) puts a box around the paragraphs the caret is in, or around all selected paragraphs. There are three types:

Type	Look
Info	Left border in the header colour of the mail, grey background. For notes and context.
Warning	A ⚠ triangle, amber border, light yellow background. For something the recipient should check.
Danger	Red border, light red background. For errors and things that must not be missed.

With the caret already inside a callout, the tool changes its type or removes the callout again.

What you see while writing

The editor shows buttons and callouts roughly as they will appear in the mail. The button colour follows the header colour of the whitelabeling live, so switching the override on or changing its value recolours the buttons in the editor at once. For the exact result, use the **Preview**.

How they are stored and sent

A template does not store the finished mail markup of a button or callout. It stores a lean marker, `<div data-mail-button>` for a button and `<div data-mail-callout="…">` with the type for a callout. The server allowlist lets both markers through and removes a callout type it does not know.

The real markup, tables with inline styles that Outlook also renders correctly, is built only when the mail is displayed: when it is sent, in the **Preview** and in the Mail Queue. All three therefore always use the current header colour of the mail.

> ℹ **A later header colour change reaches existing mail tasks.** Because buttons and callouts are built at display time, a mail task that already exists picks up a later change of the header colour for its buttons and callouts when it is shown in the Mail Queue or sent again.

Whitelabeling

By default every mail looks the same: a card in the colours of your VeloxFactory installation, read straight from the application's stylesheet. On top sits a dark header bar in the secondary dark colour of the theme (`#2D3432` in the standard theme) with the logo, directly beneath it a 4 px line in the primary colour, then the content and a footer. Table headers in the body carry the primary colour as well. Change a colour in the frontend and the mails follow, without touching a template.

When one customer or one process needs something else, a template can override parts of that frame. Each override has its own switch, and switching it off returns that part to the global default:

Override	Effect
Header colour	The colour of the header bar, instead of the dark default. Table headers in the body follow it, and so do the default background of buttons and the border of Info callouts. The line beneath the header keeps its own colour.
Header line colour	The colour of the 4 px line beneath the header, instead of the primary colour.
Hide header line	No line beneath the header, so the header bar meets the content directly.
Body colour	The background of the content area, instead of white.
Logo	A different Common Report Resource in the header, instead of the configured logo.
Hide logo	No logo at all, for a plain header.
Footer text	A different footer line. Supports the full placeholder catalog, so it can carry a trace id, a contact or a legal note.

Screenshot placeholder: `mail-template.whitelabel.png`
The whitelabeling block with the per-field switches for header colour, header line colour, hide header line, body colour, logo and footer text.

Colour and visibility of the header line show up in the preview of the edit page right away, before the template is saved, so a customer colour can be tried out against the real frame instead of guessed.

The failure notice the Job Scheduler sends when a scheduled job fails uses the same frame, dark header and primary colour line included, so both kinds of mail look alike. It is not bound to a template and has no overrides.

Preview

The **Preview** button renders the template inside the real mail frame, with example values or against an existing history record, and shows it in a dialog. It is the same rendering path the actual mail uses and the same one the Mail Queue uses to show a sent mail, so the three cannot drift apart: what the preview shows is what is sent and what is archived.

Use it after every change. It is the fastest way to catch a placeholder that resolves to nothing, a table that lost its borders or a whitelabel colour that makes the text unreadable.

The API

Mail Templates are a regular API resource under `/api/v1/mail-template`.

Endpoint	Description
`GET /mail-template/placeholders`	The placeholder catalog, optionally expanded for a given `reportConfig`.
`GET /mail-template`	List templates.
`GET /mail-template/{id}`	A single template, with its audit segment on request.

Endpoint	Description
`POST /mail-template`	Create a template. The body is sent Base64-encoded and is cleaned against the allowlist before it is stored. The description of `bodyHtmlBase64` lists the two markers for buttons and callouts, for integrations that send HTML directly.
`POST /mail-template/{id}/preview`	Render subject and body, with example values or against a history record, and report which placeholders stayed unresolved.
`PATCH /mail-template/{id}`	Partial update.
`DELETE /mail-template/{id}`	Delete a template, refused while mail tasks still reference it.

The whitelabel settings are regular fields of the resource, for example `headerLineColor` for the line colour and `hideHeaderLine` for the switch. Left empty or off, each one falls back to the global default.

Unknown placeholders in a stored template are not an error. They are returned as messages, so an integration can warn without being blocked.

AI assistants

The MCP server exposes the same operations, including the placeholder catalog. An assistant can therefore write a template that uses the real parameter names of a specific report, and preview it before anyone sends it.

Permissions

Mail Templates use the standard permission scheme and have their own **Mail Templates** section in the user editor.

Permission	Grants
`mail-template:read`	See the menu entry, the overview and a template. Also required to pick a template when sending.
`mail-template:create`	Create templates.

Permission	Grants
`mail-template:update`	Change templates, including the whitelabeling and the active state.
`mail-template:delete`	Delete templates.
`mail-template:full`	All of the above, like `global:admin`.

Uploading the graphics a template uses is a separate permission, `common-report-resource:create` and `:update`, because those files are shared with the reports.

Revision #5
Created 2026-09-20 12:05:14 UTC by Benjamin Fischer
Updated 2026-09-27 17:38:48 UTC by Benjamin Fischer