User guide

Planning and change management in one product: from what matters to your organisation, to the work that changes it. For everyone who uses DilliC, and for the administrators who set it up.

November 2026. The screens in this guide show Northwind Logistics, the demonstration organisation that ships with DilliC, signed in as its administrator. What you see depends on your role and on the modules your organisation runs, so some tabs and buttons may be missing for you. Several nouns (factor, assessment, initiative, solution, sprint, version, change request, and others) can be renamed by your organisation; this guide uses the words DilliC ships with.

CONTENTS

PART ONE · USING DILLIC

1 **Getting started**

Signing in
Starting a new organisation
The screen
Choosing where you are
Searching
Notifications, watching and mentions
Sending feedback
On your phone
Where to start, by what you do
A first pass, start to finish

2 **Your settings**

Password
Two-factor sign-in
AI assistants
Access tokens
Picture
Notifications
Language and appearance

3 **Roles: what you can do**

How roles work
The shipped roles
Administrators

4 **Overview**

5 **Strategic planning**

Factors and objectives
Proposing a factor
Voting and discussing
Accepting and rejecting
KPI readings
Refining a factor
Categories and goals
The planning diagram

6 **Assessments**

Kinds of assessment
The assessments list
Creating an assessment
Launching, reminding, closing
Taking a survey
Reading the results
How the numbers are worked out
Themes
Composites
People outside the organisation

7 **Root cause analysis**

Assertions
Agreeing and deciding
Causes and root causes
Linking initiatives

8 **Solutions and initiatives**

Initiatives
Budget and spend

KPI readings and progress
Breaking an initiative into work
Solutions
Assessing a solution
Comparing assessments
Adoption: the people the change happens to

9 Delivery

The tabs
The board
Asking questions of the work
Configuring a board
Backlog and new work
Lists
Sprints
Timeline and roadmap
A work item
Approvals on work
Risks and dependencies
Versions and releases
Change requests
Services and the change calendar
Deployments
Service desk and portal
Incidents, problems and ratings
Planning poker
Pages
Working with an AI assistant
Delivery above the teams

10 Portfolio and scaled planning

The portfolio kanban
Value streams
Planning intervals
The program board

11 Dashboards and reports

Dashboards
Gadgets
Template pages
The shipped reports
Writing your own report
AI analysis

PART TWO · ADMINISTRATION

12 Administration and modules

Who can administer
The sections
Modules

13 Hierarchy

Levels
Entities
Weights
Archiving and deleting

14 People and roles

Adding people
Giving someone a role
Editing and deactivating
Writing a role

15 Vocabulary and estimation

Renaming the nouns
The breakdown of work
Survey scales
Matrix quadrant text
Estimation and sizes
Currency

16 **Priority, components, labels and fields**

How work is prioritised
Priority and scoring scales
Components and labels
Link reasons
Requests
Custom fields

17 **Statuses, workflows and approvals**

Statuses and states
Moves
Several workflows
Statuses teams add
Approval gates
Risk assessment and routing
The change request workflow

18 **Automation**

Writing a rule
Triggers, conditions, actions
What a rule may do
The rule log

19 **KPIs and scaling**

The KPI library
SAFe levels

20 **Identity and provisioning**

Sign-in connections
Two-factor policy
AI assistants
SCIM provisioning

21 **Connections**

Version control and CI
Environments
Outbound webhooks
Apps
Two-way tracker sync

22 **Billing**

Plans and prices
Seats
AI credits
Storage
Payment and invoices

23 **Data and feedback**

Data and residency
Export and import
Feedback

REFERENCE

A The query language

B States, statuses and the default workflows

C Glossary

D The service desk by email

E Connecting Okta, Entra ID and Google Workspace

F If something goes wrong

PART ONE

Using DilliC

Everything a person using DilliC day to day needs: signing in, planning what matters, measuring where you stand, choosing what to do about it, and running the work that does it.

CHAPTER 1

Getting started

DilliC has two halves. **Planning** works out what matters, measures where you stand, and chooses what to do about it. **Delivery** runs the work that does it. Both read the same organisation, the same people, and the same figures, so a story on a board can be traced back to the goal it serves.

Signing in

1. Open the address your organisation gave you and enter your email and password.
2. If your organisation signs in through a directory such as Okta, Entra ID, or Google Workspace, a **Sign in with …** button appears once DilliC recognises your email's domain. Use it instead of a password: the directory checks who you are, and DilliC still decides what you may do.
3. If you have an authenticator app set up, DilliC asks you to **Enter your six-digit code**. A recovery code works in its place if you have lost the phone. If your organisation requires a second factor and you have none yet, you are asked to set one up before you go any further.

Forgotten your password? on the sign-in page emails you a link to set a new one. It does not apply to people who sign in through a directory; their password lives there.

An administrator adds you to DilliC and gives you a role on part of the organisation. DilliC emails you a link to choose your own password. Some organisations instead let the directory create your account the first time you sign in, or have the directory create it in advance; either way you arrive with the role your administrator chose.

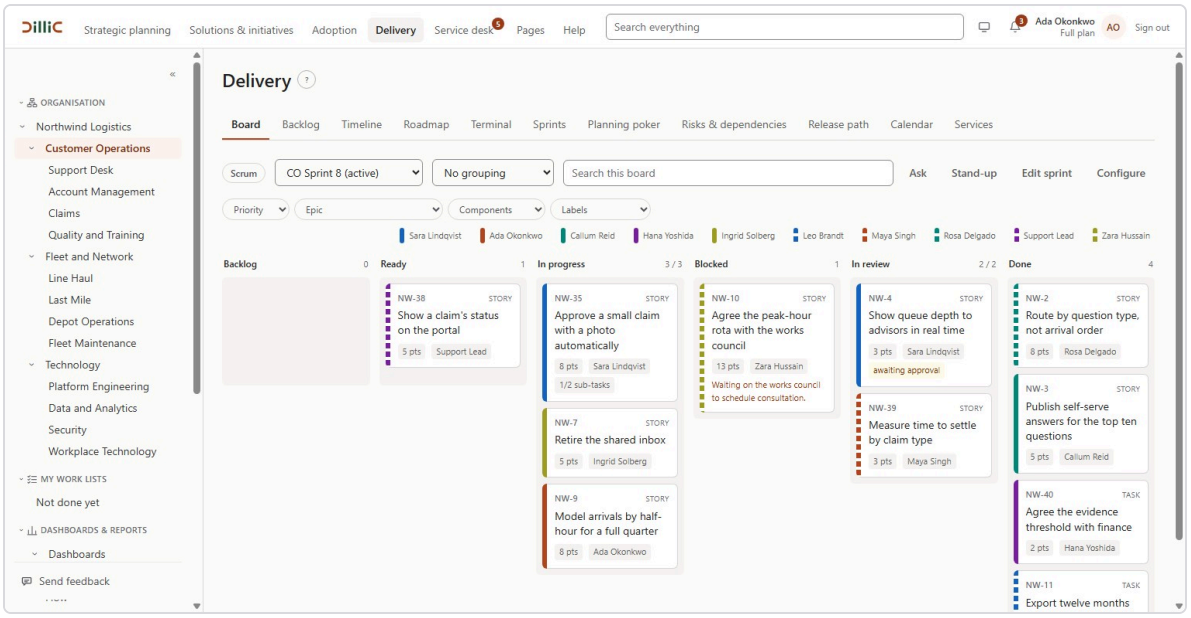
> **IF YOU ARE REFUSED**
>
> "This account has been deactivated" means an administrator switched your account off; ask them to reactivate it. "… is not set up in DilliC" after a directory sign-in means the directory knows you but DilliC does not yet: ask an administrator to add you.

Starting a new organisation

The sign-up page asks for four things: **Organisation**, **Your name**, **Email**, and **Password**. Click **Create my organisation**, and DilliC emails a confirmation link. The organisation is created only when you follow it, and you arrive signed in as its first administrator on **Admin** → **Modules**, so the first choice you make is which halves of DilliC to run (chapter 12). The starter plan is free for ten people and needs no card.

Sign up only to start an organisation of your own. To join one that already uses DilliC, ask its administrator to add you.

The screen



The header across the top, the pane down the left, and the page for whatever you chose. Here: Customer Operations' board.

The header carries the main areas, left to right. You see only the ones your organisation runs and your role reaches. At its left is DilliC's own mark, or your organisation's logo once an administrator has given the chosen entity one in the hierarchy.

TAB	WHAT IT IS FOR	CHAPTER
Overview	Where the part of the organisation you are in stands.	4
Strategic planning	The factors and objectives that matter here.	5
Assessments	Surveys of those factors, their results, and root cause analysis.	6, 7
Solutions & initiatives	What you might do, and the pieces of change it is made of.	8
Adoption	The people a change happens to: stakeholders, readiness, the plan, resistance, and adoption, by initiative. It sits before Delivery because a change is best delivered with a plan for the people it happens to.	8
Delivery	Boards, backlogs, sprints, releases, and change requests.	9
Portfolio	Where the organisation runs it: funding epics through value streams.	10
Service desk	Requests from outside, answered against promised times, from what the team knows.	9

To the right: **Search everything**, a switch between light and dark, the notification bell, and your name. Under your name is your plan ("Full plan" or "Starter plan"), or a reminder to "Set up two-factor" if you have not. Your name opens your own settings; **Sign out** is beside it.

The left pane has four groups, each of which folds away:

- **Organisation**: the tree of your organisation, company down to team.
- **My work lists**: the lists you have saved (chapter 9), wherever their work comes from. Only you see them.
- **Dashboards & reports**: your dashboards, the reports DilliC ships, the reports you have written, and, once there are any, the reports colleagues have shared.
- **Administration**: only for account administrators (part two).

Pin any row with its ☆ to keep it at hand. The « control folds the whole pane to a strip at the edge when you want the width back; resting the pointer on the strip opens it over the page. On a narrow screen the pane becomes a **Menu** button and opens as a drawer.

Most headings carry a small **(?)**. It opens a longer explanation of what the thing is and why it behaves as it does. When something in DilliC surprises you, the (?) beside it is the first place to look.

Choosing where you are

Everything in DilliC belongs to a part of your organisation, called an **entity**: a company, a division, a team, or whatever levels your organisation defined. Click an entity in the **Organisation** tree and every screen re-points to it: its factors, its assessments, its board. DilliC remembers your choice between visits.

Higher up the tree, screens gather what is beneath. A division's Delivery page shows its teams' work rolled together; a company's assessment can combine its divisions'; a report can be read "for this entity and everything beneath it". Your role follows the same shape: a role on a division reaches every team inside it.

Searching

Search everything in the header looks through work items, change requests, initiatives, pages, assessments, and solutions. Type two or more characters; every word you type must appear. Up to eight of each kind are listed. Enter opens the first result, and Escape closes the list. Archived things are included and marked "Archived", and you see only what you are allowed to open. If more matched than are shown, add a word to narrow it.

Search skips factors, KPIs, and people. Find a factor on Strategic planning, and a person wherever a person picker offers them.

Notifications, watching and mentions

The **bell** shows how many unread notifications you have, and updates as they arrive. Click one to open the work item, page, or survey it is about; it is marked read as you do. **Mark all read** clears the count. What reaches you, and where, is up to you (chapter 2).

Watching a work item means hearing when it changes. Use **Watch** at the top of an item; it reads **Watching - tap to stop** while you are, and its tooltip lists who else follows the item. You start watching automatically when you create an item, are assigned it, or comment on it. Once you stop watching, DilliC does not start you again.

Mentioning someone means typing `@` in a comment or description and picking them. They are notified. DilliC offers only people who can open the item, and refuses a mention of anyone who could not, because a mention would tell them about something they cannot see.

Sending feedback

Send feedback at the foot of the left pane reports a bug or suggests something, from any screen:

1. Choose **Something is wrong** or **A suggestion**.
2. Say it **In a line**, and add **Details** if they help.
3. Optionally **Capture this screen**, **Attach files**, or paste an image: up to six files, 10 MB each.
4. **Send**. The screen you were on, your browser, window size, and language go with it.

Reports go to the DilliC team's service desk. Your administrators can see what people in your organisation have sent under **Admin** → **Feedback**.

On your phone

The DilliC app for iPhone and Android aims to do everything the web app does, on phones and tablets. It has five tabs (home, plan, deliver, insights, and more) and uses the same addresses as the web, so a link in an email or a notification opens the same screen on either.

- It signs in with a password and second factor, or through your directory in a browser sheet. Signing out revokes the phone's access.
- It receives push notifications when "My phone" is ticked in your notification settings.
- It reconnects its live updates whenever you come back to it.
- Dialogs open as sheets from the bottom, a (?) opens on tap, and downloads go through the phone's share sheet.

Where to start, by what you do

This guide describes every screen, and nobody needs all of it. Find yourself in the table and read those sections first. The rest can wait until you meet it.

YOU	READ FIRST	THEN, WHEN YOU NEED IT
Were asked to answer a survey	Signing in (this chapter); Taking a survey (chapter 6)	Reading the results (chapter 6)
Do work on a team's board	The screen (this chapter); The board, and A work item (chapter 9)	Notifications (chapter 2); Lists, and Asking questions of the work (chapter 9)
Lead a team's delivery	The same, with Backlog and new work, and Sprints (chapter 9)	Configuring a board (chapter 9); Dashboards and reports (chapter 11)
Lead planning for part of the organisation	A first pass, below; then chapters 5 to 8 in order	Overview (chapter 4); Adoption (chapter 8)
Answer requests on the service desk	Service desk and portal (chapter 9)	Incidents, problems and ratings (chapter 9)
Approve work or changes	Approvals on work, and Change requests (chapter 9)	Services and the change calendar (chapter 9)
Administer the account	Chapters 12 to 14: modules, hierarchy, people	The table that opens part two says who usually does the rest

A word you do not know is likely to be in the glossary (Appendix C), which also explains the words DilliC borrows from agile delivery and from IT. When something does not behave as you expect, look in Appendix F.

A first pass, start to finish

DilliC is built around one loop: say what matters, measure it, ask why, decide what to do, do it, and measure again. Here is one pass through that loop, as Customer Operations at Northwind Logistics might make it. Each step names the screen, and the chapter that describes it in full. A first pass with a few factors and one initiative teaches more than reading ahead.

1. **Choose where you are.** In the **Organisation** tree, click the part of the organisation you are planning for: here, Customer Operations. Every screen now shows its factors, its assessments and its board.
2. **Say what matters.** On **Strategic planning**, click **Propose a factor/objective**. Write the factor in neutral words, "The speed of responding to customer questions", and its objective, "Reply to every question within X time by Y date". Colleagues vote on whether it applies, and a leader chooses **Accept**. A first round needs only a few accepted factors. (Chapter 5.)
3. **Measure where you stand.** On the **Assessments** tab, click **New assessment**, choose **Current state**, give it a title such as "Mid-year review", and **Create**. Then choose **Launch** in its Actions menu. Everyone with a role here is told a survey is waiting, and each answers it in one sitting. When enough have answered, choose **Close accepting responses**. (Chapter 6.)

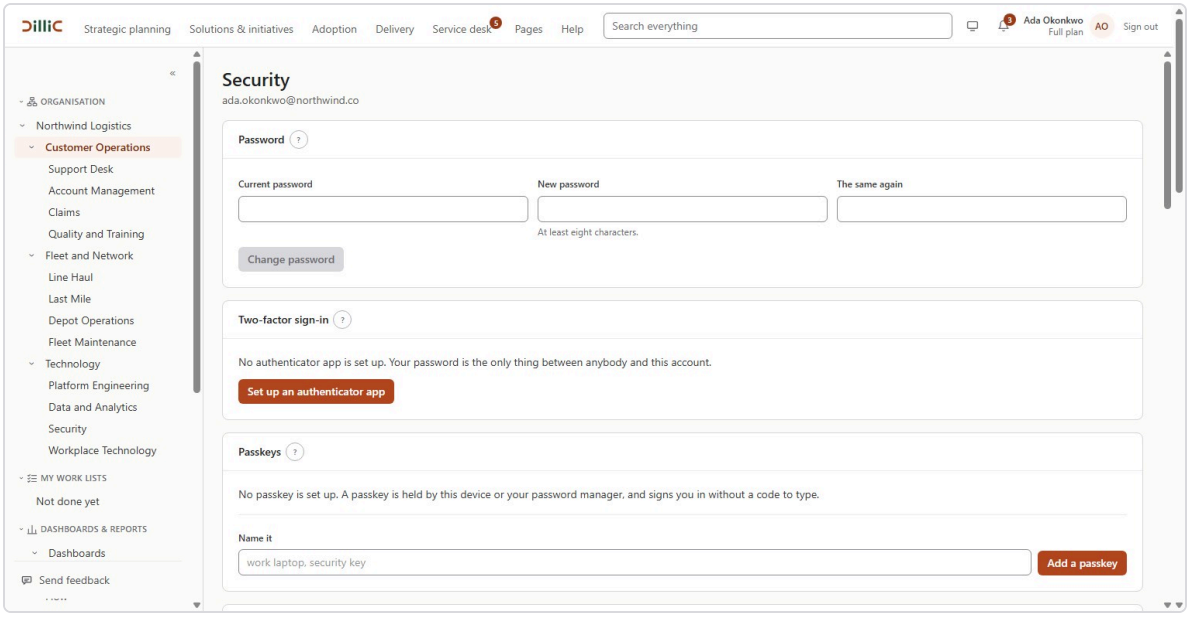
4. **Read the results.** Open the assessment. Each lettered circle on the matrix is a factor: weak on the left, strong on the right, opportunity above, threat below. Begin with the factors on the weak side and the ones people disagreed about, and read their comments before deciding anything. (Chapter 6.)
5. **Ask why.** Switch the results to **Root cause analysis**. Click **New assertion** and state what is wrong with one factor: "Customers wait too long for a first reply during the afternoon peak". Under it, **Add a cause beneath this** answers "why is that the case?". Keep asking until a cause has nothing beneath it. That one is a root cause, and a leader accepts it. (Chapter 7.)
6. **Decide what to do.** On **Solutions & initiatives**, click **New initiative** and name the piece of change, "Shift Handover Redesign", with a budget and the measure that will show whether it worked. Link it to the root cause it addresses. If there are rival answers, propose each as a solution, assess each as a desired future state, and compare them. (Chapter 8.)
7. **Turn it into work.** On the initiative, **Add epic** creates the first large piece of work beneath it. On **Delivery**, open the **Backlog**, click **New item**, and add stories under the epic, each with an estimate in points. (Chapters 8 and 9.)
8. **Do the work.** On a board that runs sprints, open **Sprints**, click **Plan a sprint**, pull stories into it from the backlog, and **Start** it. People drag their cards across the board as the work moves on. **Complete** the sprint when it ends. A Kanban board has no sprints: work is pulled from the backlog as there is room for it. (Chapter 9.)
9. **See whether it worked.** Record readings against the initiative's measure and the objective's. The **Benefits** tab on Strategic planning then says whether the objective is on course and which initiatives are moving it. Later, run a second current state and compare it with the first. (Chapters 5 and 8.)

> **THE LOOP, READ BACKWARDS**
>
> Open any work item and choose its **Trace** tab. It reads the same chain in the other direction: from the story, through its initiative and the root cause it addresses, to the objective and the goal. Where the line breaks, it says what is missing.

CHAPTER 2

Your settings

Click your name in the header. The page, titled **Security**, shows your email at the top and holds everything that is yours alone.



Your settings: password, two-factor sign-in, and access tokens, with more below.

Password

Enter your **Current password**, a **New password** of at least eight characters, and **The same again**, then **Change password**. If you sign in through a directory this card does not apply: change the password there.

Two-factor sign-in

A second factor means a password alone is not enough to get in as you.

1. **Set up an authenticator app** and scan the code with an app such as Google Authenticator, Microsoft Authenticator, or 1Password.
2. Type the **Code from the app** and **Turn it on**.
3. DilliC shows your **recovery codes** once. Keep them somewhere safe and apart from the phone; each gets you in once if the phone is lost. Then **Done**.

With it on, a **Current code** lets you **Replace the app** (a new phone) or get **New recovery codes**. **Turn two-factor off** asks for your password, and is not offered if your organisation requires a second factor. If you lose the phone and the codes, an administrator can reset your second factor; you keep your password and set up a new app.

AI assistants

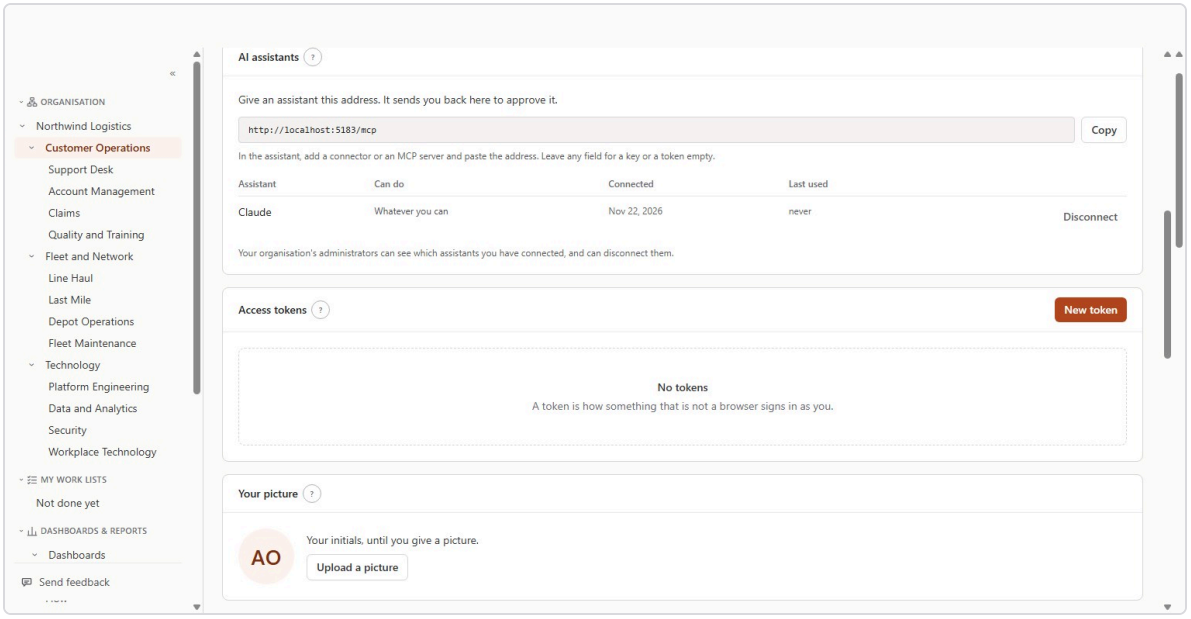
An AI assistant such as Claude or ChatGPT can work in DilliC for you: find work, read it, and change it. You give it one address and approve it once. It never sees your password, it does only what you may do, and the audit trail records its work under your name and names the assistant beside it.

1. Under **AI assistants**, **Copy** the address. It ends in `/mcp` and is the same for everyone in your organisation.
2. In the assistant, add a connector (some assistants say "MCP server") and paste the address. Leave any field for a key or a token empty.
3. The assistant opens DilliC in your browser. Sign in if you are asked to. The page says which assistant is asking, who you are signed in as, and where your answer will be sent.
4. Choose what it may do. **Whatever I can** gives it your reach. **Only what I tick** keeps it to the permissions you tick, and such an assistant can never administer the account.
5. **Connect**. The browser returns to the assistant, which is now connected.

> **BEFORE YOU CONNECT**
>
> The name on the page is the assistant's own word; the address under **Your answer will be sent to** is what DilliC checked. **Refuse** if you did not start the connection yourself, or do not know that address.

Then ask in plain words: "What is assigned to me?", or "Add a story called Review the returns policy." The assistant finds the right part of DilliC, and most assistants ask your leave before they change anything. What it reads and changes is bounded by your roles, exactly as in the web app.

The list under the address shows each connected assistant, what it can do, when it connected, and when it was last used. **Disconnect** stops one at once; to let it back in, connect it again. A connection nobody uses for ninety days lapses by itself. A tool with nowhere to send you, such as a script, takes an access token instead.



AI assistants: the address to give one, and the assistants you have connected.

Your organisation's administrators can see which assistants you have connected, and can disconnect them. They can also turn assistants off for the whole organisation; the card then says so, and no assistant can be connected.

Access tokens

A token lets a script or tool act as you without your password: a CI job reporting build status, or the terminal launcher described in chapter 9.

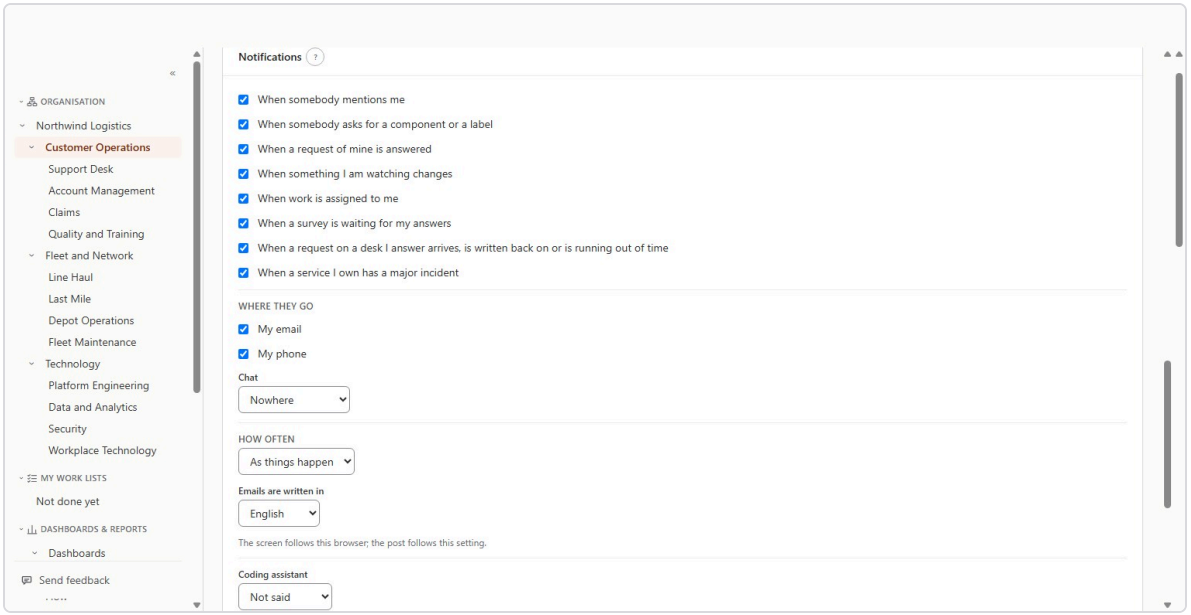
1. **New token**. Say **What it is for** (for example "CI build status").
2. Optionally set when it **Expires**. Left empty, it never does.
3. Optionally tick **Scopes**. With none ticked, the token can do whatever you can. Ticking any limits it to those, and a scoped token can never administer the account.
4. Copy the token when it is shown. It is shown once only; then **I have it**.

The table lists each token with what it can do, when it was last used, and when it expires. **Revoke** stops one at once; revoked tokens stay in the list for the record. Signing in on the phone app creates a token too, marked "Phone".

Picture

Your picture is shown beside your name wherever you appear. Choose a file, or remove it to go back to your initials.

Notifications



What reaches you, where, and how often.

What. Tick the kinds you want:

- When somebody mentions me
- When something I am watching changes
- When work is assigned to me
- When somebody asks for a component or a label (administrators)
- When a request of mine is answered
- When a survey is waiting for my answers
- When a request on a desk I answer arrives, is written back on or is running out of time

DilliC leaves out the kinds that belong to parts of it your organisation does not run.

Where. Any of:

- **My email**.
- **My phone**: push notifications to the app.
- **Chat**: Slack or Microsoft Teams. Create an incoming webhook in the chat tool for the channel you want, paste its address, and **Save the address**. It is never shown again; **Forget it** removes it.

How often. As things happen, or **Once a day** at an hour you choose, in your own time zone. A day with nothing in it sends nothing.

Emails are written in one of DilliC's eight languages, separately from the language of the screen. **Coding assistant** remembers which AI assistant the terminal launcher should open for you.

Language and appearance

Language offers English, Français, Español, Português, Deutsch, العربية, हिन्दी, and 中文, each named in its own language. Dates and numbers follow it. The choice is kept in this browser and carried to your account when you sign in. **Appearance** is System, Light, or Dark, for this browser only.

CHAPTER 3

Roles: what you can do

How roles work

An administrator gives you a **role** on an entity. It applies to that entity and to everything beneath it, so a leader of a division leads its teams without being added to each one. You can hold different roles in different places, one per entity: a contributor on your own team and an observer on another division, say.

A role is a named set of rights. The four DilliC ships with are below; your organisation may have changed them or written its own (chapter 14). If a button you expect is missing, the usual reason is that your role does not carry that right, or that the part of DilliC it belongs to is switched off.

The shipped roles

ROLE	MEANT FOR	CAN
Leader	"Runs planning for an entity and everything beneath it."	Everything except the portfolio rights: accept factors, launch assessments, decide root cause analysis, govern initiatives, manage sprints and versions, configure boards, approve work and change requests.
Contributor	"Shapes factors and takes part in assessments, without governance rights."	Propose, vote on, and refine factors; answer assessments and see results; propose and answer in root cause analysis; read initiatives and reports; view, edit, and move work; read and write pages; log time; answer service desk requests; vote in planning poker; read and edit change requests; write planning-interval objectives; manage risks.
Participant	"Answers surveys and reads results."	Read and vote on factors; answer assessments; answer in root cause analysis; read work.
Observer	"Read-only access to results and reports."	Read factors, assessment results, root cause analysis, initiatives, and reports.

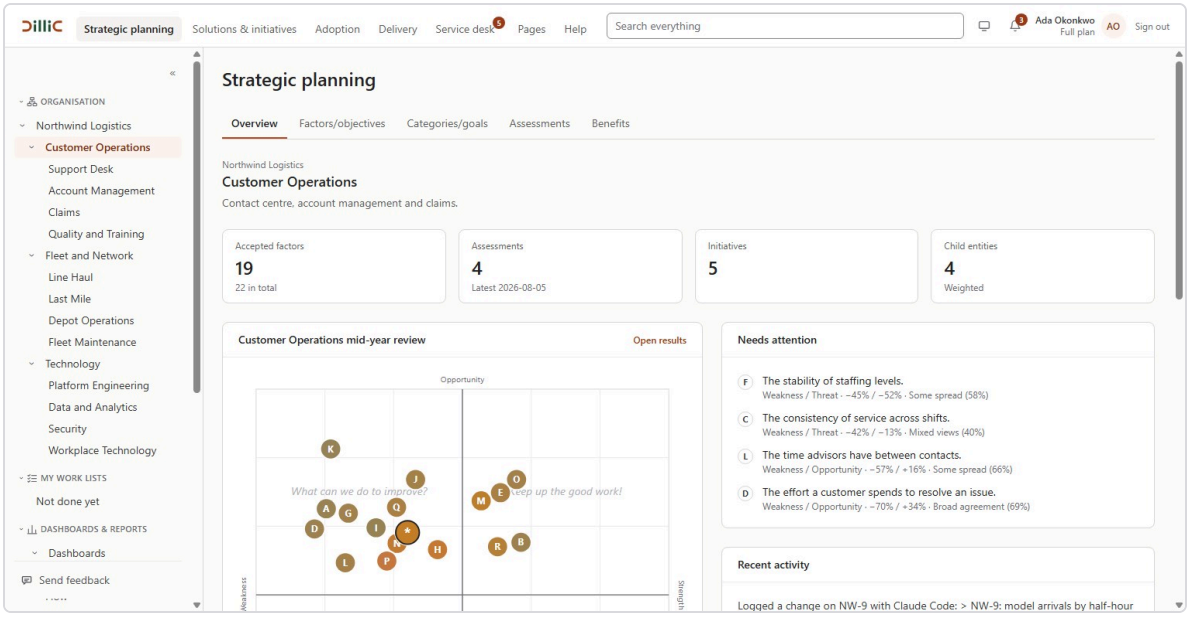
Administrators

Being an **administrator** is not a role but a tick on a person: "Administers the account". An administrator holds every right everywhere, sees the Administration group in the left pane, and can reset another person's second factor. An organisation always keeps at least one.

CHAPTER 4

Overview

Overview answers "where does this part of the organisation stand?" for whichever entity is chosen in the pane. It is the first tab of Strategic planning, and it appears once an assessment has been run here: until then there is nothing to overview, and the screen opens on the factors.



Customer Operations at a glance.

At the top: the entity's place in the organisation (for example "Northwind Logistics / Customer Operations"), its name and description. Then, depending on your rights:

- **Accepted factors**: how many, and how many in total.
- **Assessments**: how many, and the date of the latest.
- **Initiatives**: how many.
- **Child entities**: how many, and whether they are weighted.

The largest card is the **latest assessment's matrix**, with **Open results**. With no assessment yet it suggests the first step: accept a set of factors, then run a current state assessment.

Needs attention lists the four factors that scored worst on that assessment, with their quadrant, scores, and how far people agreed. **Recent activity** lists the last eight things that happened here and who did them. When an assistant or an access token did one in a person's name, the line says so: *Ada Okonkwo, through the assistant Claude*. Below come the entity's children, with their weights, and its first five initiatives with their status.

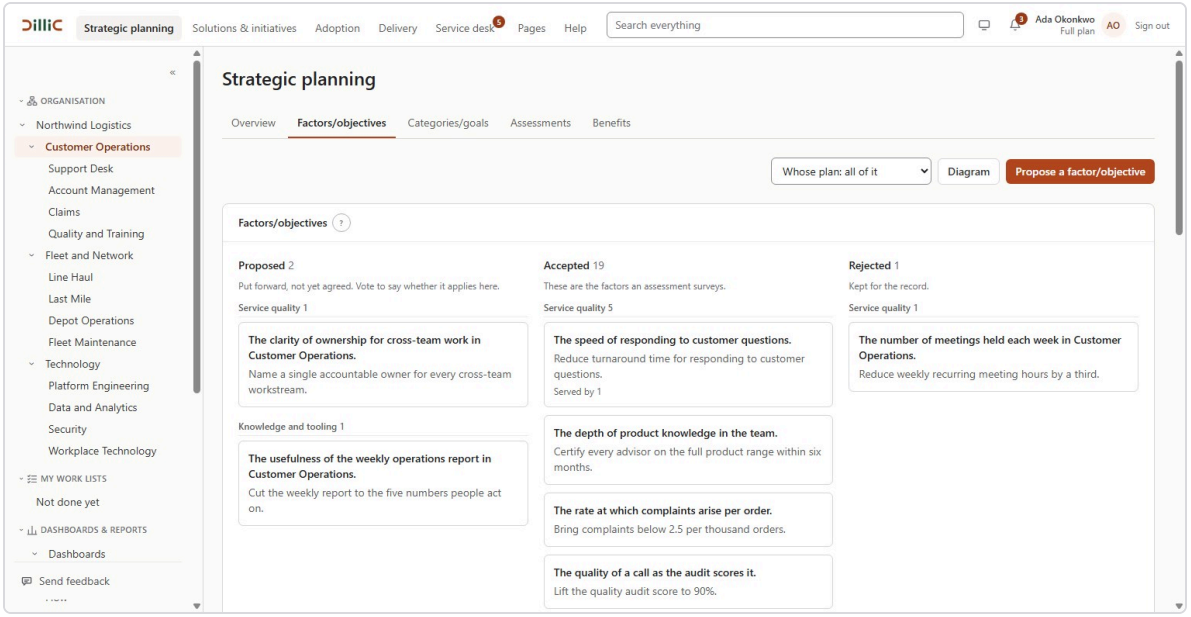
CHAPTER 5

Strategic planning

Planning starts by agreeing what matters to a part of the organisation, worded so it can be scored, and what "better" would look like for each.

Factors and objectives

A **factor** is something that matters here, worded neutrally so that people can say whether it is a strength or a weakness: "*The speed of responding to customer questions.*" Its **objective** says what you intend to do about it: "*Reply to every question within X time by Y date.*" Behind the objective can sit a **KPI** (what gets counted), a **target**, and a **date**, which together make it SMART. A factor and its objective are one record, because a factor with no objective cannot be acted on and an objective with no factor cannot be scored.



Strategic planning. Factors sit in three lanes and are grouped by category.

Strategic planning shows the chosen entity's factors in three lanes, each with a count, and grouped under their categories (with "No category" last):

- **Proposed**: put forward, not yet agreed. Vote to say whether it applies here.
- **Accepted**: agreed. These are the factors an assessment surveys.
- **Rejected**: kept for the record.

Strategic planning is one screen with four tabs: **Overview** (chapter 4), **Factors/objectives**, **Categories/goals**, and **Assessments** (chapter 6). The tab is in the page's address, so you can send one to a colleague. **Diagram** opens the planning diagram.

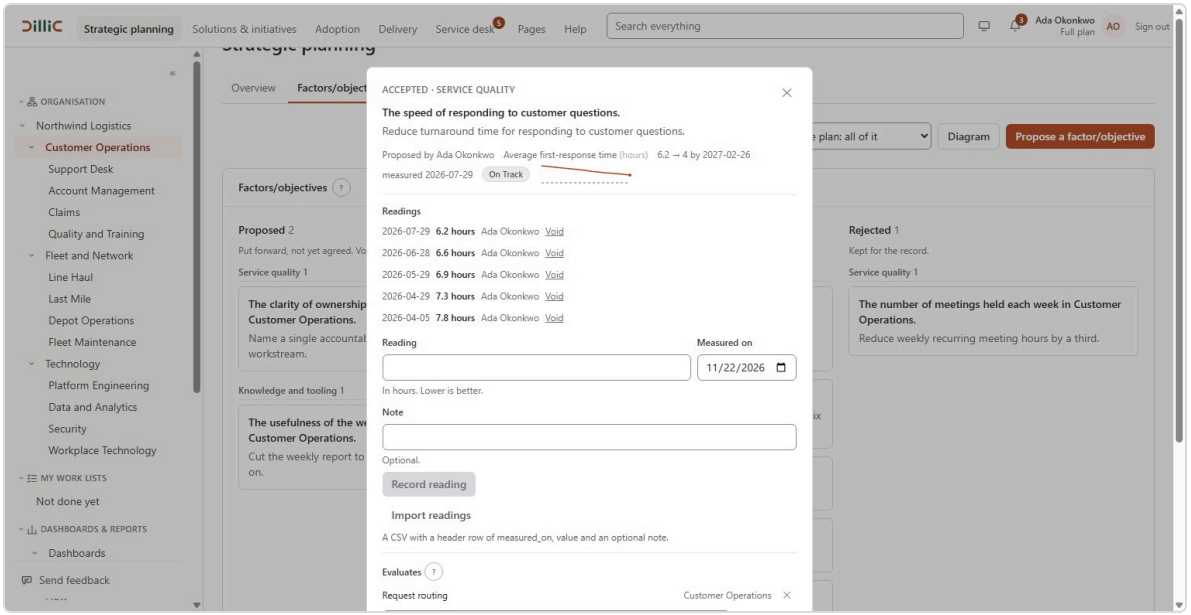
Proposing a factor

1. Choose your entity, open **Strategic planning**, and click **Propose a factor/objective**.
2. Write the **Factor** in neutral words: not "slow responses" (already a judgement) but "the speed of responding". A neutral factor can be scored either way.
3. Write the **Objective**: the specific, relevant part of SMART. Use placeholders (X, Y) for the numbers; the numbers belong in the target.
4. Choose a **Category**, or leave it ungrouped.
5. Where the organisation takes its surveys the traditional way (chapter 15), **Proposed as** offers Strength, Weakness, Opportunity, and Threat, or "Not said — the survey asks". People vote on the factor and the kind together, and the survey opens on it, where the person answering can still change it.
6. Optionally complete the objective under **What completes the SMART objective**: a **KPI — measurable** from the organisation's library, an **Initial target value — attainable**, and **By when — timebound**.
7. **Propose**. The factor lands in Proposed. Somebody who may accept factors also has **Jump to Accept** beside it, which proposes and accepts in one step, skipping the vote on whether it applies.

A target must sit inside the KPI's range and is rounded to the KPI's decimal places; the hint under the box says what the range is.

Voting and discussing

Click a factor to open it.



A factor opened: who proposed it, its objective, the votes, and the discussion.

The dialog shows its status and category, what it was proposed as, who proposed it, and, when there is a KPI:

- the KPI and unit, the current value, and the target "by" its date;

- a trend: *reached*, *on track*, *off track*, or *not enough readings*, worked out from a straight line through the last five readings, with a small chart;
- where the same KPI is measured beneath, the figure rolled up from those entities;
- how many times the target has been revised.

On a proposed factor, **Does this factor apply here?** offers the organisation's agreement scale (by default from "Completely agree" to "Completely disagree"). Your vote saves as soon as you pick it; voting again replaces it. Once you have voted, **Why?** takes a comment explaining it, saved with **Add comment**. The person who proposed a factor does not vote on it.

The votes add up to **Applicability**: a percentage where "completely disagree" is 0%, neutral is 50%, and "completely agree" is 100%, shown with the number of votes. Each comment shows the answer its writer chose.

Accepting and rejecting

A leader chooses **Accept** or **Reject** in the dialog. The votes inform the decision but do not make it: a leader can accept a factor people were lukewarm about, or reject a popular one, and DilliC writes every change of status to the history. An assessment surveys only accepted factors.

KPI readings

When a factor has a KPI, people with the right to refine factors can record readings: **Reading** (inside the KPI's range), **Measured on** (not in the future), and an optional **Note**, then **Record reading**. A reading entered by mistake can be **Void**ed. Readings move the current value and the trend; they never move the target.

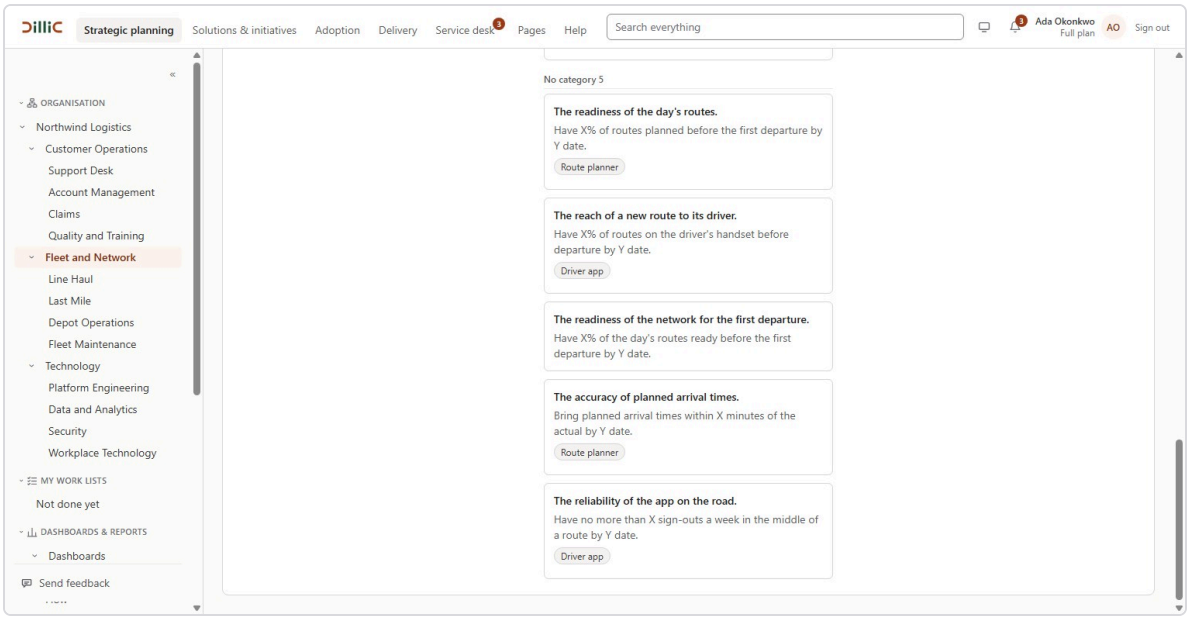
Import readings takes many at once from a CSV file whose header row names `measured_on` and `value`, with an optional `note`, and one reading a line, dated `YYYY-MM-DD`. A reading already on record, the same day and value, is skipped, so a file sent twice writes nothing the second time. One bad line refuses the whole file and names the line. A script can post readings too: give it a personal access token narrowed to the right to refine factors or edit initiatives, and have it post to the readings endpoint in the API reference. Initiatives import the same way.

A measure/KPI whose readings **come from the services** (chapter 19) is not typed in: the dialog says *Read from the services each month*, and each reading names the services it was read from. A factor may say which services it **Evaluates** (chapter 9, Services): optional, because a factor about culture or skills names none. One that names some is read off those alone rather than off every service its part of the organisation runs, and the assessment's results can be read a service at a time. Changing the services re-reads the months on record off the new set; a reading somebody voided stays voided.

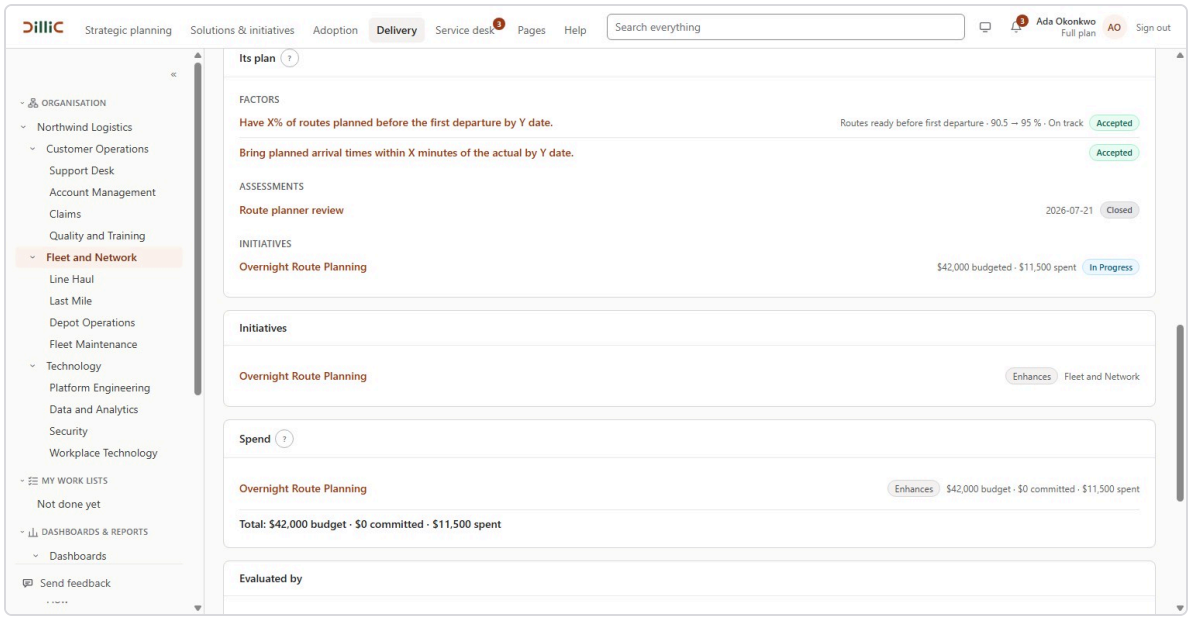
Planning one service at a time

A part of the organisation that runs several services (chapter 9) need not blur them into one plan. Where it runs any, the form for a factor carries **Belongs to**: the part itself, or one of its services. So do the forms for an assessment, an initiative, a risk, and a PI objective. Nothing obliges you to choose a service, and the two kinds of plan stand together: what belongs to no service is the part's own plan.

- Each list can be read by **Whose plan** it is: everything, the part's own, or one service's. Each row that belongs to a service names it.
- A factor's card shows its KPI read across the part's services **By service**, beside the part's own figure and never added into it.
- Where a part runs more than one service, each may be given a **Weight** on its form (chapter 9): how much it counts among them. Weights are relative, so 2 beside 1 counts twice, and they need total nothing. They are used in a composite of the services' assessments, and in a KPI whose rule is weighted, only when every service being combined has one. Leave them blank, or weigh only some, and the services count equally.
- The parts above count each factor and each initiative once, whichever plan it is in. Take care with a KPI that sums: a part that types a total which already includes its services' figures is counted twice from above, as a parent that includes its children's would be.
- A factor that belongs to a service also **Evaluates** it, and that one cannot be removed from the list.
- A factor's service may be changed until an assessment has scored it. An initiative, a risk, and a PI objective may be moved at any time. An assessment stays with what it was taken of.
- What you may do to a service's plan is what you may do on the part that runs it.



Fleet and Network's factors: the division's own, and those that belong to its route planner and its driver app.



The route planner's page: its plan, with its factors, the assessment taken of it, its initiative and the money.

Refining a factor

Refine changes a factor's wording, objective, category, kind, or measure. It opens on **What kind of change is this?**, three choices spelled out, and nothing beneath is live until one is chosen, because the answer decides whether past results can still be compared with future ones:

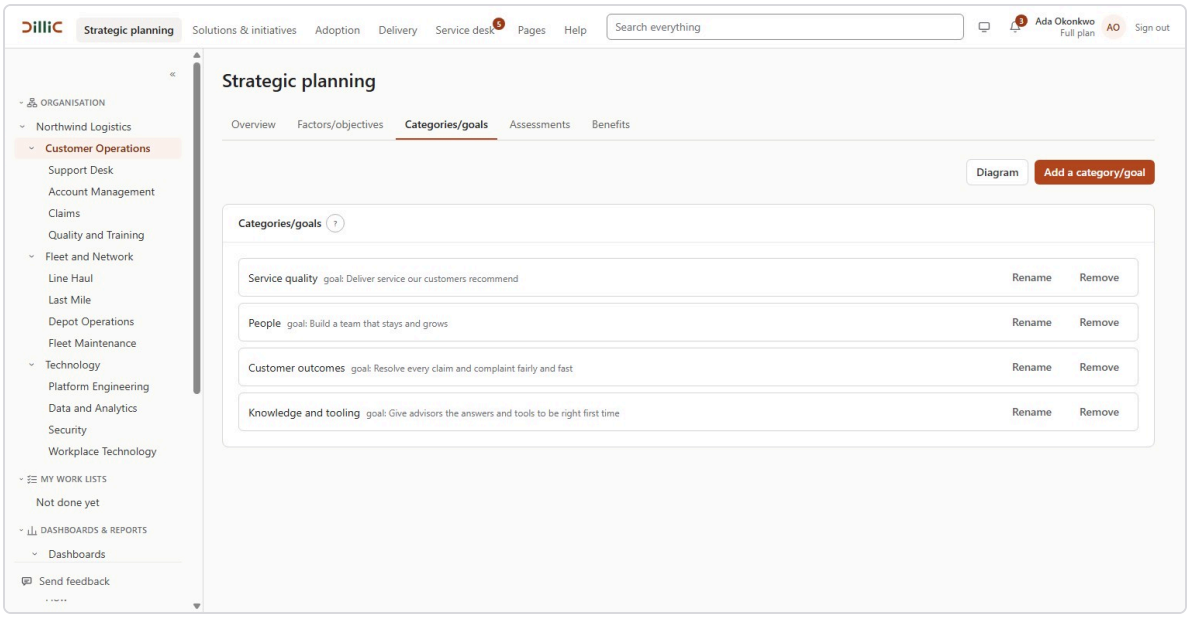
KIND	MEANING	EFFECT
Cosmetic	Wording tidied up. Nobody would have answered differently.	Comparison survives. The KPI and target cannot change.
Partial reset	Only the KPI target moved.	Comparison survives; the change is logged, and the new target is saved beside the old one.
Full reset	The factor now asks something different.	The chain breaks: the next assessment becomes a new baseline, and earlier ones are no longer comparable.

Say **Why**; it is recorded in the factor's **Change history**, with the date, the kind of change, and who made it. Moving a factor to another category, or changing what it is proposed as, is never a reset.

> **A RESET ASKS AGAIN**
>
> A partial or full reset says people would have answered differently. So on every survey still taking answers, open or paused, that factor's answers are cleared, the question is re-read from the factor, and everybody who had answered is told and asked that one question again; the rest of their answers stand. Closed rounds keep what they were asked.

Categories and goals

A category groups factors ("Service quality", "People") and names the goal they serve ("Deliver service our customers recommend"): the category is the neutral word, the goal the aspiration. On the **Categories/goals** tab, **Add a category/goal** opens a form, and **Save** adds it; if you leave the goal empty it reuses the category's word. Each category has **Rename** and **Remove**. Removing a category that holds factors leaves them ungrouped; file them again with Refine.



Categories and goals, which the factors are filed under.

Benefits: are the initiatives paying off?

The **Benefits** tab, last on Strategic planning, answers the question the rest of the screen sets up: each accepted objective, the initiatives meant to move it, and whether they are moving it. An objective shows its measure/KPI against the target and date, its trend, and a word for its course - **on course**, **off course**, **reached**, **not enough readings**, or **no target**. Beneath it, each initiative that serves it shows whether it is *adopted* (all its adoption targets met, from the Adoption tab) and *effective* (its own measure/KPI meeting its target), and a verdict that puts the two together: adopted and effective, adopted but not yet effective, effective but not yet adopted, neither yet, or not measured.

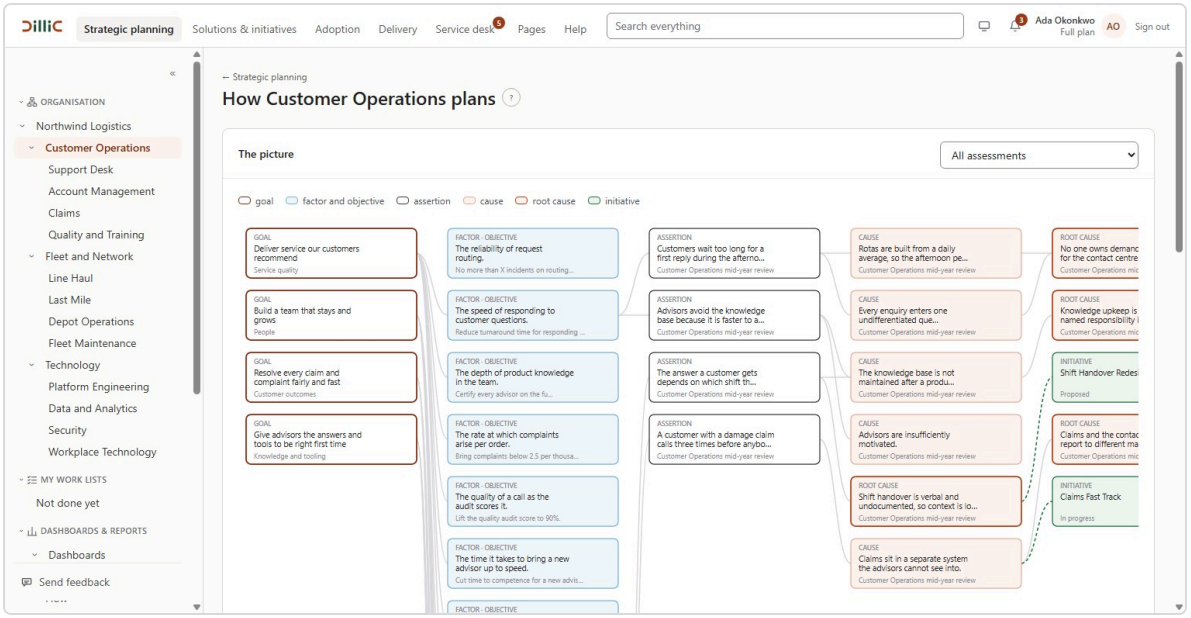
A sentence on each objective reads them together. *On course through its initiatives* is the plan working. *Its initiatives are working, but the objective is off course* means the bets are paying out but not moving the number - worth asking whether they were the right bets. *Off course, and some of its initiatives are behind* points at the initiatives to chase. *Nothing serves this objective yet* is a finding in itself. Tick **Include the parts beneath** to read the objectives of every part of the organisation below the one you are in.

You say by hand which initiatives serve which objective, because the link is a claim somebody makes and can be held to: **Link an initiative** in a factor's **Served by** section, or **Link an objective** under **Serves objectives** on an initiative. Suggestions come first, with the reason: a root cause the initiative is linked to explains this factor, it is measured by the same measure/KPI, or its solution's desired future state assessed the factor. An initiative can serve an accepted objective on its own part of the organisation or any part above it; linking

takes the right to edit the initiative. Without Adoption, or without the right to see it, only the effective half shows. Completed initiatives stay listed, marked archived, because a benefit is usually realised after the work is done.

The planning diagram

Diagram draws the whole chain on one page, left to right: goal → factor and objective → assertion → cause and root cause → initiative. Links into initiatives are drawn dashed. **The picture** can show every assessment's analysis or only one's.



The planning diagram for Customer Operations.

Click any node, or tab to it and press Enter, to open it: an initiative opens its own dialog, and anything else offers to take you to it on Strategic planning or in the analysis. If you cannot read root cause analysis, the picture stops at the factors and says so. The diagram scrolls sideways on screen and fits the page when printed.

CHAPTER 6

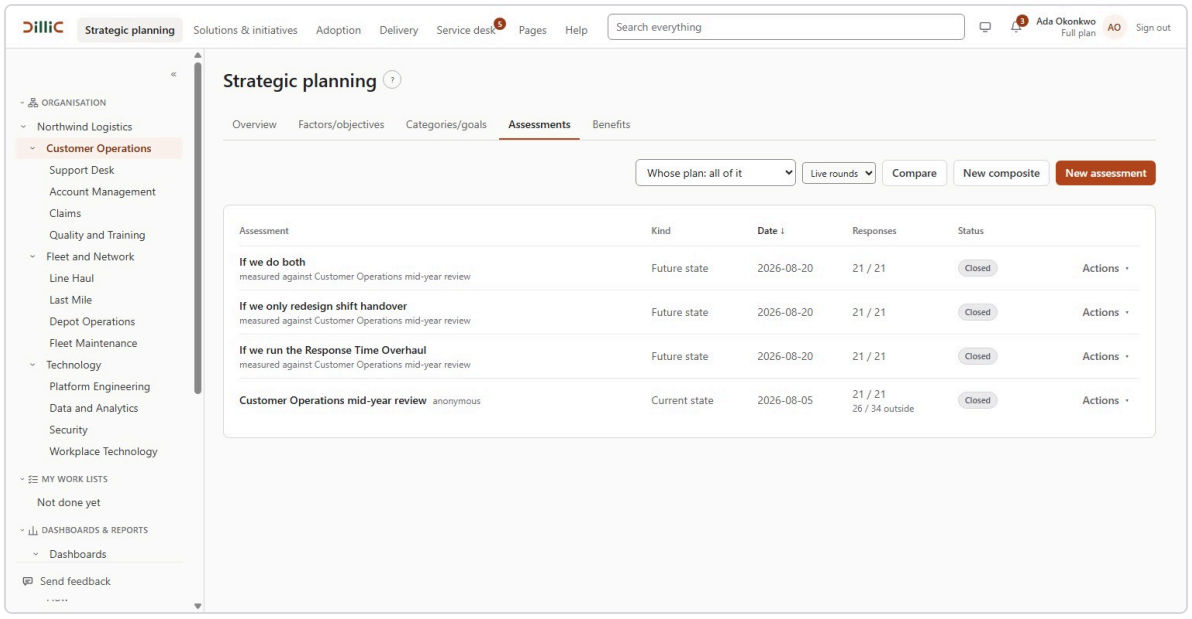
Assessments

An assessment asks people to score the accepted factors, and draws the answers as a matrix: strengths and weaknesses across, opportunities and threats up and down.

Kinds of assessment

KIND	WHAT IT ASKS
Current state	How things are today. It freezes the entity's accepted factors, as they are worded now, into a survey.
Desired future state	How things would be if chosen initiatives were delivered by a date. It reuses a current state's factors, so the two can be compared. A desired future state is how solutions are compared before one is chosen (chapter 8).
Composite	Children's assessments rolled up at a parent. Nobody answers it directly.

The assessments list



Customer Operations' assessments: one current state and three solutions' future states.

The list shows each assessment's kind, date, responses (completed out of invited, with people outside counted on a second line) and status: draft, open, paused, cancelled, or closed. Assessments are a tab of Strategic planning. Future states say which current state they are measured against; anonymous ones say so. The list shows live assessments by default; switch to archived or all to see older rounds. Every column sorts.

Creating an assessment

1. **New assessment**. Choose **Current state** ("The entity as it is today") or **Desired future state** ("The entity assuming chosen initiatives are delivered").
2. Give it a **Title** ("Mid-year review") and optional **Instructions for participants**.
3. Set the **Assessment date** (today by default).
4. For a future state, also: **Implemented by**, the current state it is **Measured against**, and the **Initiatives assumed delivered**. With two or more initiatives, name the solution they make up.
5. Decide on **Anonymous responses** (on by default). When it is on, DilliC stores no author against a response at all, and tells participants which setting applies before they answer.
6. Optionally write a thank-you for people outside who answer. Then **Create**.

The assessment starts as a **draft**. Its questions are fixed from this moment. Later changes to the factors reach it in two ways only: a factor refined as a partial or full reset re-asks that one question (chapter 5), and a round paused to rework factors reads them afresh when it resumes. The factors are lettered A to Z, then a to z, so an assessment holds at most 52. With no accepted factors there is nothing to assess, and DilliC says so.

> **THE ANSWER SCALES**
> Every assessment is answered on the organisation's own scales, set by an administrator (chapter 15), so that every assessment can be compared with every other. No single assessment can change them.

Until it is closed, **Edit**, in the row's Actions menu, changes its title and instructions, and the wording of the invitation and thank-you sent to people outside. Once the round is closed, only the thank-you can change.

The Actions menu

Every row has one **Actions** menu, offering only what the round's status and your role allow. A draft offers Edit, Launch, and Delete; a launched round Edit, Take survey, Invite outsiders, Pause to rework factors, Cancel assessment, Remind invitees, and Close accepting responses; a paused one Resume assessment and Cancel assessment; a closed or cancelled one Reopen and Archive; an archived one Bring back.

Launch	Opens the survey. Everyone with a role on this entity, or on any entity above it, becomes a participant and is told a survey is waiting.
Take survey	Offered to you while it is open and you have not answered.
Invite outsiders	Sends it to people outside the organisation (below).
Pause to rework factors	Stops answers coming in while you change the factors. The people asked stay asked and their links stay theirs. **Resume assessment** opens it again with the factors as they are by then: reworded ones read as reworded, and a factor accepted meanwhile is added under the next letter.
Cancel assessment	Stops short of closing, keeping whatever came in. A cancelled round can be reopened.
Remind invitees	Nudges everyone who has not answered, except anyone reminded about this survey in the last hour, and says how many were reminded. It is greyed out once everybody asked has answered.
Close accepting responses	Stops taking answers.
Reopen	Takes answers again after closing or cancelling.
Delete	Removes a draft, after asking. Nothing else can be deleted: once launched, a round is a record, and archiving takes it out of sight.
Archive	Takes a closed or cancelled assessment out of the live list; **Bring back** restores it. Archived rounds stay searchable and comparable.

Pausing, resuming, cancelling, reopening, and reminding each open a dialog with a **Subject** and a **Message**, both optional. If you leave both empty, DilliC makes the move and emails nobody, except for a reminder, which then says what it always said. If you write something, it goes to everybody invited: people inside through their notifications and email preferences, people outside by letter, never to the person pressing the button.

An open or paused survey cannot be archived. DilliC also archives an old current state by itself once a newer closed one has been the entity's picture for a while (two weeks, unless an administrator set otherwise for the entity).

Taking a survey

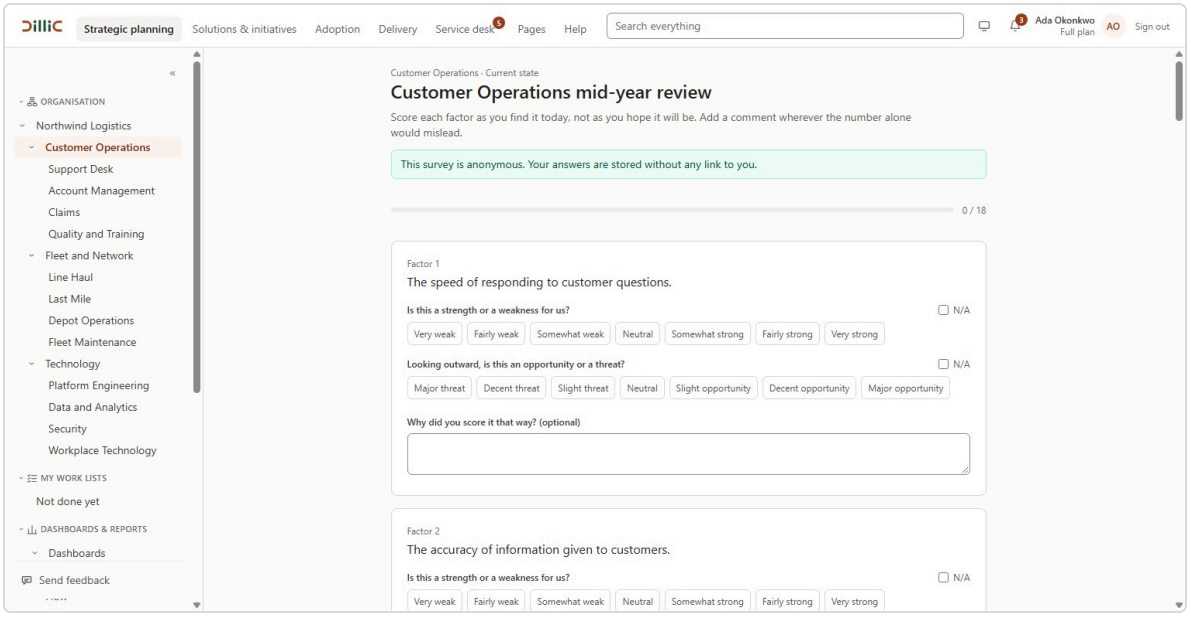
When a survey is waiting for you, you are notified, and **Take survey** is in the row's Actions menu. The survey shows its instructions and whether it is anonymous. Each factor asks two questions:

- *Is this a strength or a weakness for us?* From "Very weak" to "Very strong".
- *Looking outward, is this an opportunity or a threat?* From "Major threat" to "Major opportunity".

Where the organisation takes its surveys the traditional way (chapter 15), each factor asks one question instead, *Is this a strength, a weakness, an opportunity or a threat for us?*, and then how much of one, on that kind's three rungs. There is no middle, and one **N/A** skips the factor. A factor put forward as a kind opens on it, and you can change it. The answer is scored the same way either way: that axis, with the other left out.

If a question does not apply to a factor, tick **N/A** beside it instead of choosing a score. That question is left out of the factor's score rather than counted as neutral: its position on that axis comes from everybody who did answer. Tick both to decline the factor altogether - the comment box then asks why, and what you write is kept with the results. The factor's detail says how many marked each question N/A.

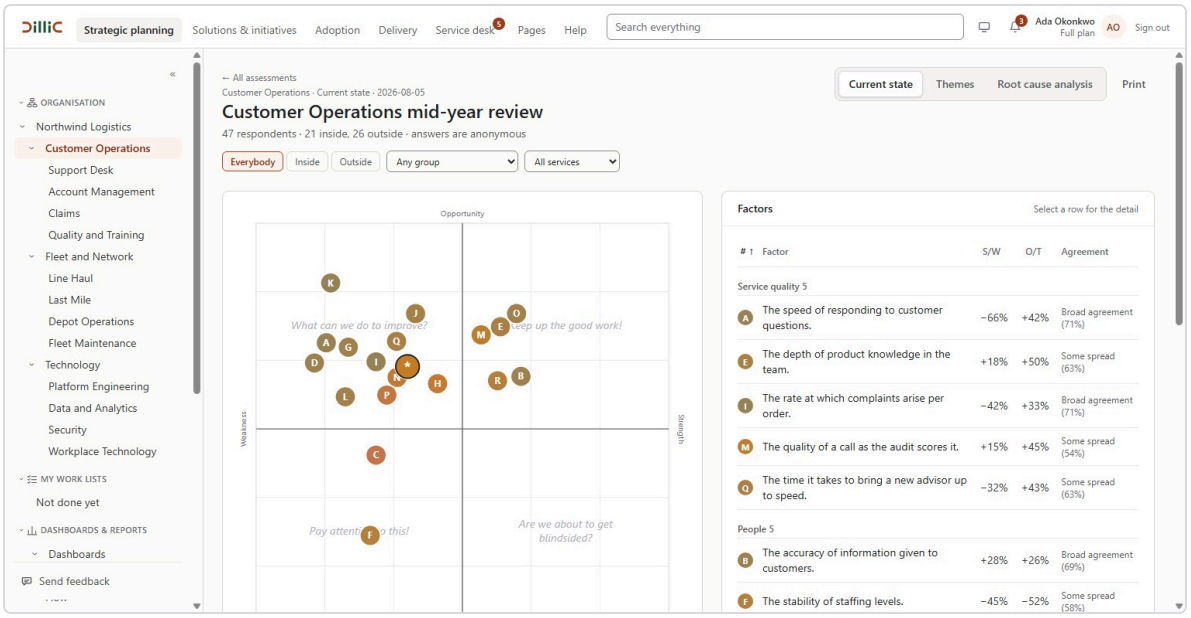
Add a comment wherever the number alone would mislead. A bar at the top counts how many factors you have answered on both questions, counting an N/A as an answer; **Submit answers** waits until all of them are. A future state asks you to answer as though its initiatives had been delivered by its date.



A survey. The banner says whether your answers are anonymous.

> **ANSWER IN ONE SITTING**
>
> The page holds your answers until you submit, and saves no draft: leaving the page loses them. Once submitted they cannot be changed, with one exception: if a factor is reset while the survey is still running, you are told, and opening the survey again shows that one question, with a note that the rest of your answers stand. Otherwise opening it again says you have already answered and offers the results.

Reading the results



Results. Each lettered circle is a factor; ∗ is the combined score.

The header gives the number of respondents, how many were inside and outside the organisation, and whether answers are anonymous. The matrix:

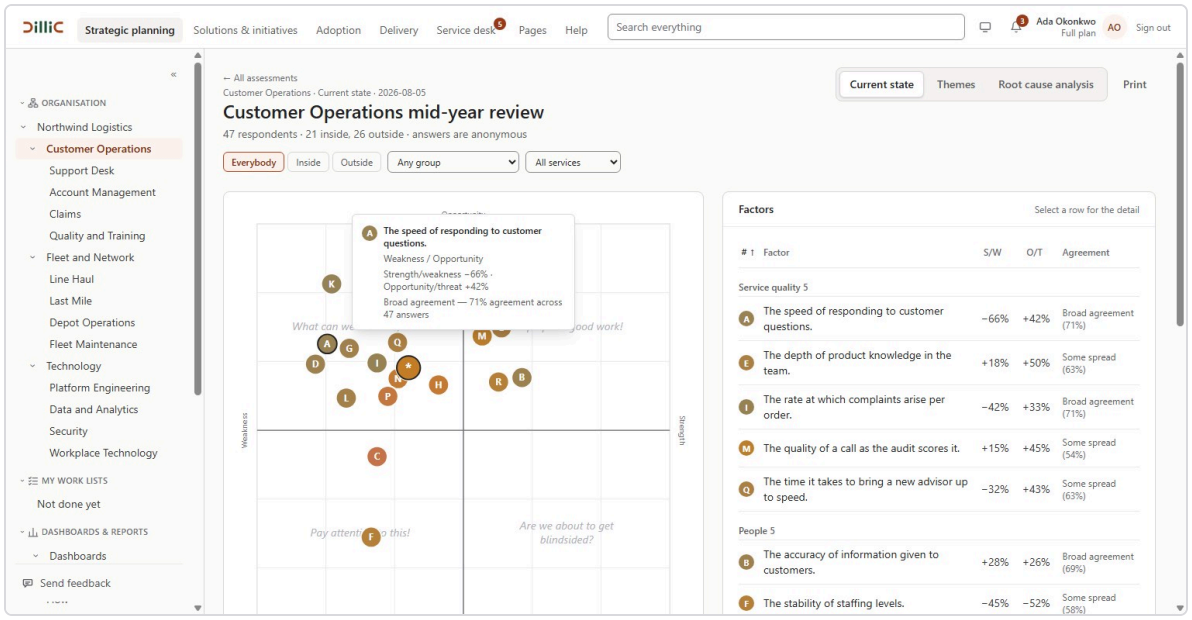
- **Across**: weakness on the left, strength on the right.
- **Up and down**: opportunity above, threat below.
- Each quadrant carries its own question: by default *What can we do to improve?* (weak, but an opportunity), *Keep up the good work!*, *Pay attention to this!* and *Are we about to get blindsided?*
- Each factor is a lettered circle, coloured by how far people agreed about it. The ∗ is the combined score.

The table beside it gives each factor's strength/weakness and opportunity/threat as percentages and its agreement in words and numbers, so nothing depends on seeing colour. Factors are grouped under their categories; every column sorts.

When people outside answered, **Everybody**, **Inside**, and **Outside** re-read the whole page for one audience, and a group menu narrows it to one group of outside respondents. The choice is kept in the page's address, so you can send a colleague the exact view.

Where the factors name the services they evaluate (chapter 5, KPI readings), a service menu narrows the matrix and the table to the factors about one service: how the respondents judged checkout, not only the team. The combined score stays the whole survey's.

Click a factor's mark or row for its detail: its two scores (as a percentage and on the −3 to +3 scale), its agreement and number of answers, and under **More**, how the answers were spread across the scale and every comment, each with the answers its writer gave. Where the spread is wide, DilliC adds that the spread matters more than the average and that you should read the comments before acting.



A factor's detail, from its mark on the matrix.

The toggle at the top switches to **Themes** and to **Root cause analysis** (chapter 7). **Print** gives a clean copy of any of them.

How the numbers are worked out

Every answer is a number from −3 to +3 (or −3 to +3 without 0, if your organisation removed Neutral).

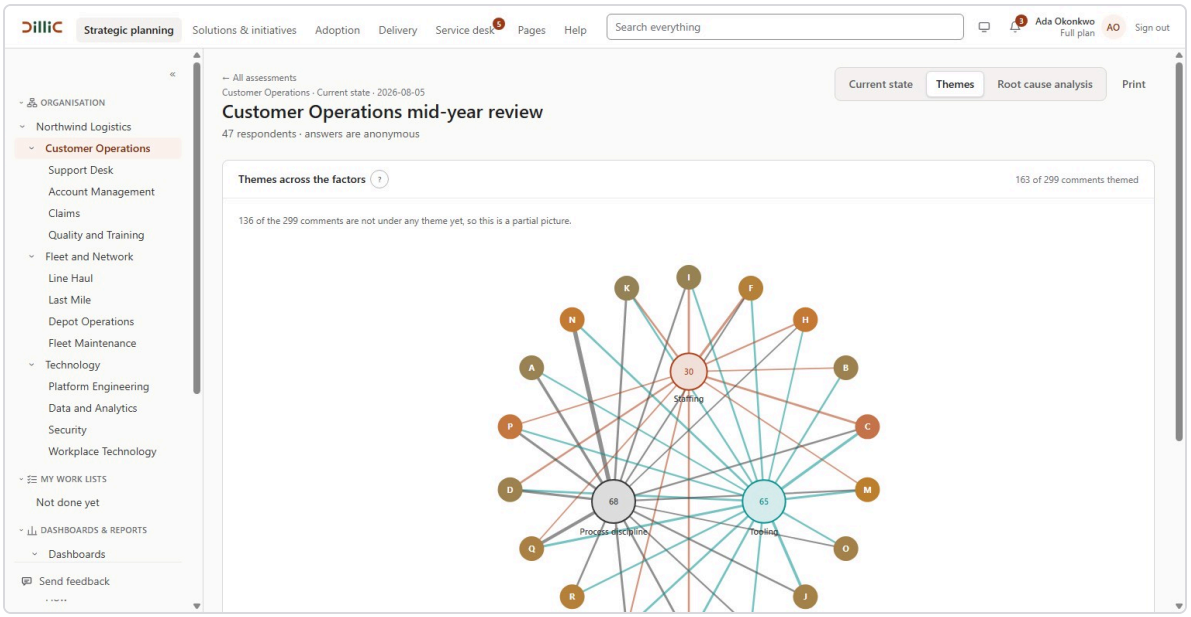
- A factor's position is the **average** of its strength/weakness answers and of its opportunity/threat answers.
- The **percentage** shown is that average as a share of the scale's end: an average of −1.5 shows as −50%.
- **Agreement** comes from how spread out the answers were: 100% means everybody answered alike, 0% an even split. It is labelled:

AGREEMENT	LABEL
above 83%	Strong agreement
66–83%	Broad agreement
50–66%	Some spread
33–50%	Mixed views
33% and below	Sharply divided

The **combined score** pools every answer to every factor, so factors more people answered count for more. A score of exactly zero is drawn on the strength and opportunity side.

Themes

Themes group what people wrote. Anyone who can see results makes them by hand: open a factor's detail, press **More**, create a theme with **New theme**, and tick the theme on each comment it fits. A comment can carry several themes.



Themes across the factors: each hub a theme, each letter a factor.

The **Themes** view draws them: each hub is a theme, sized by how many comments carry it, joined to the factors those comments were about; hovering shows one theme's connections and clicking a factor opens its comments. A table lists each theme with how often it was raised, the factors it runs through, and whether it is mostly about weaknesses, strengths, or both. The header says how many comments are themed so far.

Composites

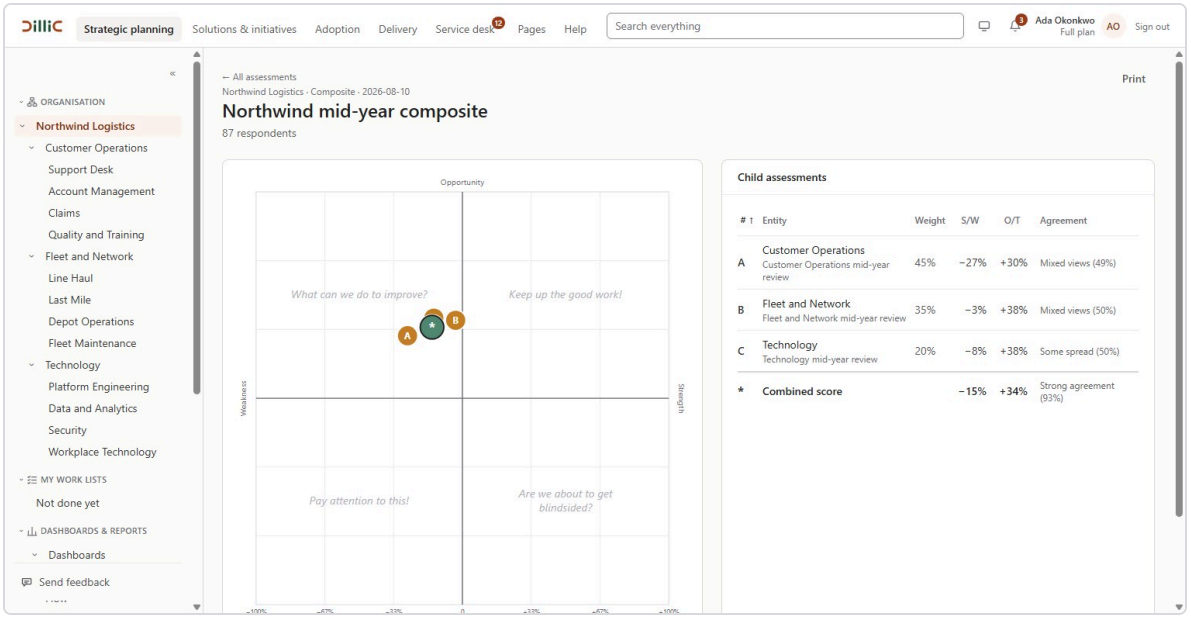
At an entity with children, **New composite** rolls their assessments up into one:

1. Give it a **Title** and **Date**.
2. Tick at least two assessments from entities beneath this one. Choose rounds taken around the same time.
3. **Build composite**.

Each child contributes its combined score as one mark. If every chosen child's entity has a weight (chapter 13), the roll-up leans towards the heavier ones; if any is unweighted, all count equally, and the form tells you which applies. A composite's agreement shows how far the children agreed with each other. It is created closed, and composites can themselves be combined further up.

An assessment may be of one service: choose it under **Belongs to** when you create a current state assessment, and the round asks about that service's factors alone. A future state assessment is of whatever its baseline is of. The list shows the part's own rounds and its services' together, and **Whose plan** narrows it.

Once two of a part's services have been assessed, **New composite** offers **Combine**: the parts beneath this one, or the services this part runs. A composite of services names each mark for its service, and leans towards the heavier ones where every service chosen has a weight; otherwise each counts equally. It cannot mix the two kinds, but it is the part's own, so a composite further up can take it.



Northwind's composite: the three divisions, weighted 45/35/20.

People outside the organisation

Customers, partners, or anyone else outside can also answer an assessment: up to 100 per survey on the free plan, and 1,000 plus 100 per seat on the full plan. They need no account and see only the survey they are sent.

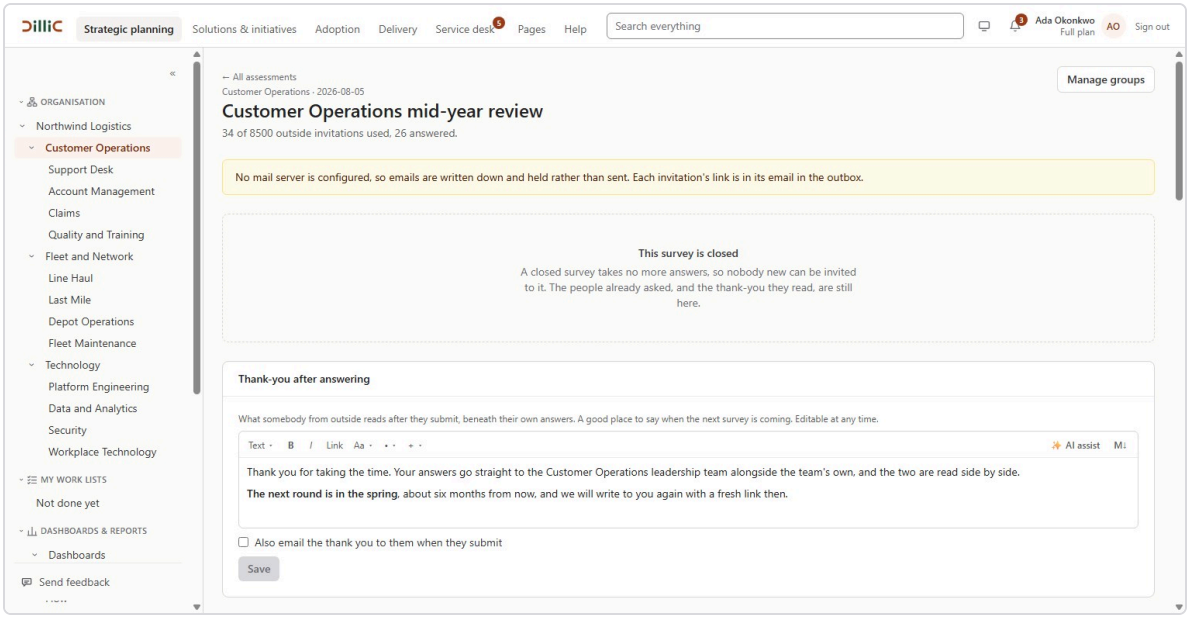
Keeping groups

They are kept in the organisation's **Groups**, under **Administration → Groups** (chapter 14), so they can be asked again: **Add by email** takes addresses one per line or separated by commas, with an optional name first (`Dana Ruiz <dana@example.com>`), and reports how many were added, already there, or unreadable. Each person can be corrected or removed; an address cannot be changed once its owner has answered a survey. A survey can invite any group owned by its part of the organisation or a part above it.

Inviting

1. Launch the assessment first. In its Actions menu, choose **Invite outsiders**.
2. Tick **Groups**, or type addresses under **Or by address**, or both. Only the people outside the organisation in a group are invited; people inside it answer through the launch, and the result says how many were left out.
3. Adjust the invitation's subject and wording if you like, then **Send invitations**. The letter is headed with the organisation's logo where the entity has one.

Each person gets one email with their own link. The result says how many were invited, already invited, or unreadable; people who unsubscribed are left out and named. A survey may invite 1,000 people plus 100 for every seat the organisation holds.



Invitations: how many are used, the thank-you, and everyone invited.

The invitations page lists everyone invited, with their groups, whether the email was sent, could not be sent, bounced, is waiting, or was withdrawn, and whether they answered. For each person:

- **Send again**: a new link; the old one stops working.
- **Correct address**, for one that bounced.
- **Withdraw**, before they answer.
- **Unsubscribe**.
- **Anonymise** (administrators): removes their name and address everywhere and stops their links. Their answers stay, linked to nobody. Anonymising cannot be undone.

Thank-you after answering is shown to people outside when they submit, and can also be emailed to them. It can be changed at any time, even after the survey closes.

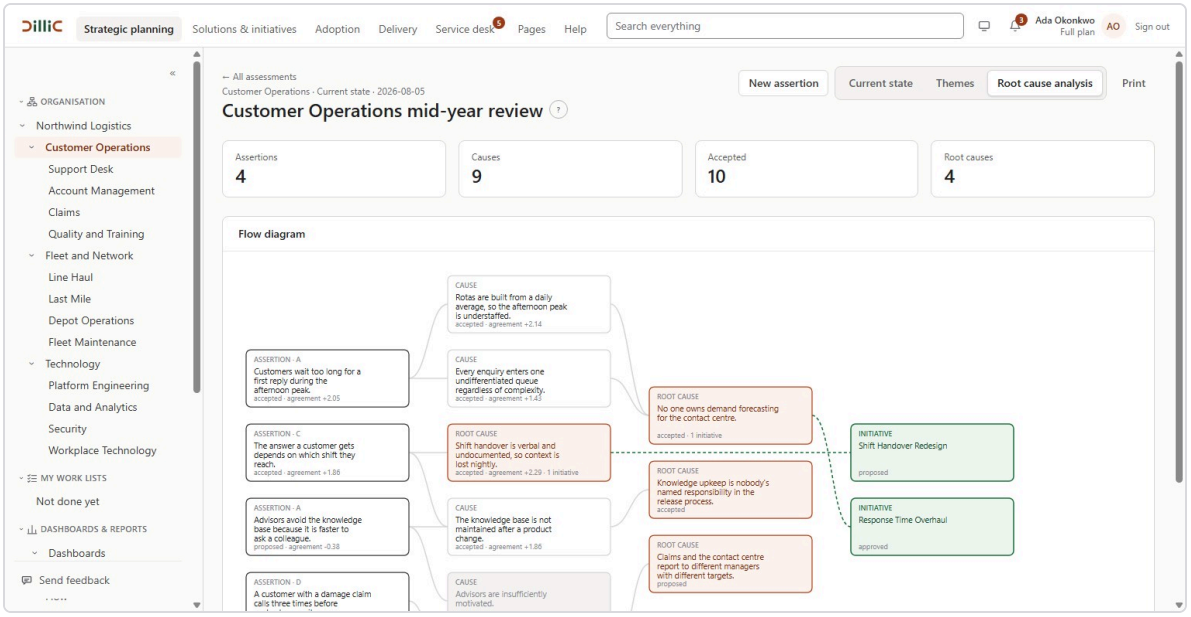
What they see

Their link opens the same survey, question for question, with the same N/A boxes. Answers from outside are never anonymous, and the page says so: the organisation can see what each person said. After submitting they see their own answers and the thank-you. A preferences page lets them unsubscribe (or ask to be emailed again) or **Delete my details**, which removes their name and address and keeps their answers unattributed.

CHAPTER 7

Root cause analysis

Results say where you stand. Root cause analysis asks why, and keeps asking until nothing is left underneath. Each assessment has one analysis, opened from the toggle on its results.



The flow diagram: assertions on the left, causes, root causes, and the initiatives that address them on the right.

The page counts assertions, causes, accepted nodes, and root causes, and draws them as a **flow diagram** from each factor's assertions to the causes beneath them and the initiatives linked on the right. **Print** fits the diagram to the page.

Assertions

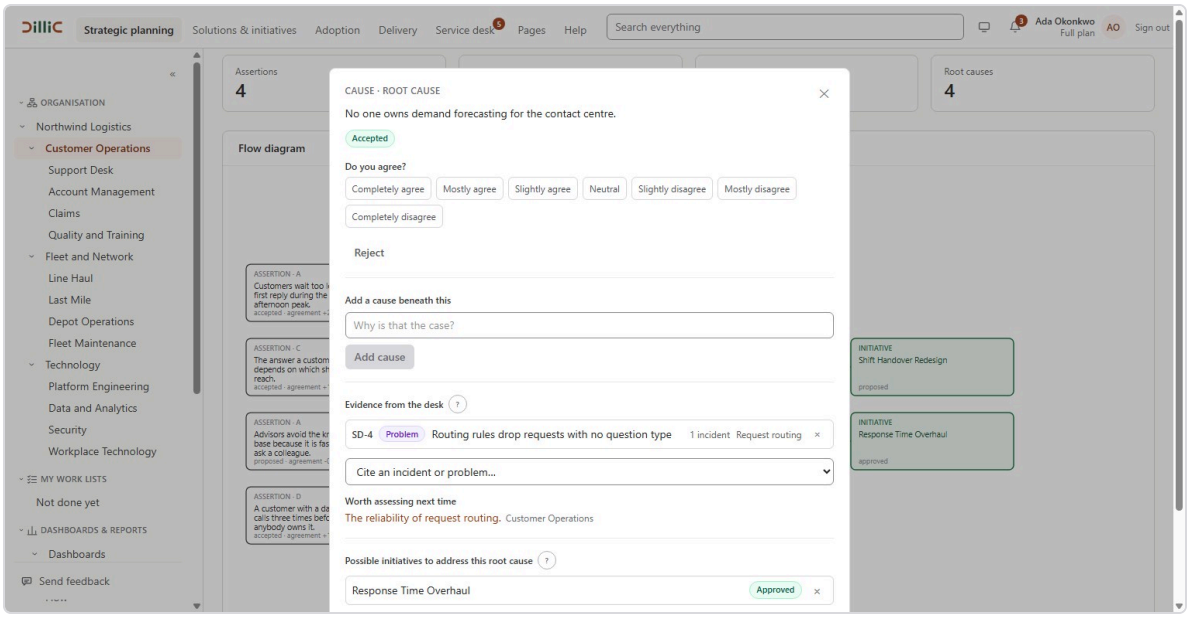
An **assertion** is a claim about one factor that people can agree or disagree with: "*Customers wait too long for a first reply during the afternoon peak.*" Use **New assertion**, choose **About which factor**, write the assertion, and **Propose**. Everything in an analysis hangs from an assertion; a cause never attaches straight to a factor.

Agreeing and deciding

Click any node to open it. It shows what it is about, its status (proposed, accepted, or rejected), and how far people agree so far. **Do you agree?** offers the same agreement scale as factor votes; answering again replaces your answer. A leader then chooses **Accept** or **Reject**, informed by the answers but not bound by them.

Causes and root causes

Beneath an assertion or cause, **Add a cause beneath this** answers *"Why is that the case?"*. Causes are agreed and decided the same way, and can have causes of their own. A cause that is not rejected and has nothing beneath it is a **root cause**.



A node opened: its agreement, the evidence from the desk, and what addresses it.

Evidence from the desk

An assertion says something is wrong, and a cause says why; the service desk holds the record of it going wrong. Under **Evidence from the desk**, anyone who may propose cites an incident, or the problem behind several, against the open node: **Cite an incident or problem…** offers what the desk holds on the services this assessment's factors evaluate (or, where none names a service, on this part of the organisation's own services), newest first, as far as you may read the desk. A problem says how many incidents stand behind it; × stops citing one.

Because a request names the services it was about and a factor names the services it evaluates (chapter 5), the citation also says what is **worth assessing next time**: the factors that evaluate the cited services and this assessment did not ask about, and the cited services no factor evaluates at all. That is the loop from operations back to planning.

Linking initiatives

On an **accepted** root cause, a leader can link the initiatives that address it under **Possible initiatives to address this root cause**, and unlink one with ×. That link is how an initiative answers for something the organisation agreed was wrong; it shows on the planning diagram, on the initiative's trace, and in reports.

If a root cause is later rejected, or a cause is added beneath it, it stops being a root cause and its initiatives are released; the dialog names them. The initiatives themselves stay as they are. Initiatives address causes, never assertions.

CHAPTER 8

Solutions and initiatives

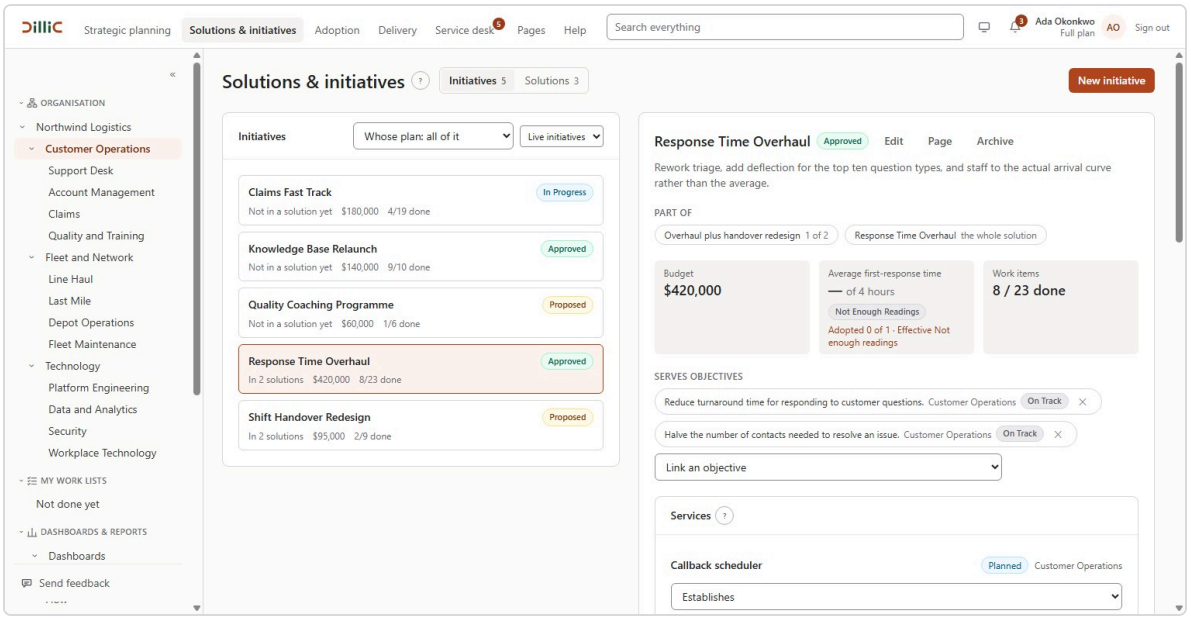
An **initiative** is a piece of change with a budget and a measure: "*Shift Handover Redesign.*" A **solution** is one way of answering the assessment, made of one or more initiatives. Solutions compete; the one you choose becomes the plan, and its initiatives become the work.

Solutions & initiatives switches between the two lists, each with a count and a live/archived/all filter, and shows whatever you select on the right.

Initiatives

New initiative asks for a **Name**, an optional **Budget**, the measure/KPI it is **Measured by** with its **Target** (both optional), and a **Description**. Editing one adds:

- **Status**: Proposed, Approved, In progress, Complete, or Cancelled.
- **Measured by**: a measure/KPI from the organisation's library, with a target and the value achieved. This says whether the initiative is *effective*; its adoption targets say whether it is *adopted* (chapter on Adoption). The card shows both.
- **Serves objectives**: the accepted objectives this initiative is meant to move, on its own part of the organisation or above. **Link an objective** adds one, suggestions first; the Benefits tab on Strategic planning reads them together.
- **Services**: the services this initiative **establishes**, **enhances**, or **retires** (chapter 9, Services), added from any team's services. One it establishes may still be *planned*. A measure/KPI read from the services reads the ones it establishes or enhances. Once the initiative is in progress, a service none of its changes has touched says *No change request touches it yet*: the plan and the delivery disagree, until a release is aimed at the right change.



An initiative: the solutions it is part of, its budget, KPI, and work done, and the spend and KPI readings recorded against it.

The detail shows the solutions it is part of (or that none has claimed it yet: work nobody has said is worth doing), then its budget, its KPI with a trend, and how many of its work items are done. An initiative marked Complete or Cancelled archives itself after two weeks; **Archive** does it sooner. **Delete** is refused while anything hangs off the initiative, and the refusal names what.

Budget and spend

Record money against an initiative under **Spend**:

1. Choose the **Kind**: **Actual** (spent), **Commitment** (promised, such as a signed purchase order), or **Forecast** (expected).
2. Enter the **Amount** (in the initiative's currency) and the day it happened **On**.
3. Optionally give the **Reference**: the finance system's own number for the line, such as a purchase-order or invoice number.
4. Optionally describe it, then **Record spend**.

The line above the form reads the budget, what is committed and spent, and what **remains**: the budget less commitments and actuals. Forecasts are shown beside it and never subtracted. Where the entity has a cost per hour, the labour of the hours logged on the initiative's work is shown as "derived". A line entered by mistake can be voided; a negative amount records a correction, like a credit note. Dates cannot be in the future.

An initiative with child initiatives beneath it also shows what they are budgeted at and have spent, and warns if they are allocated more than it holds.

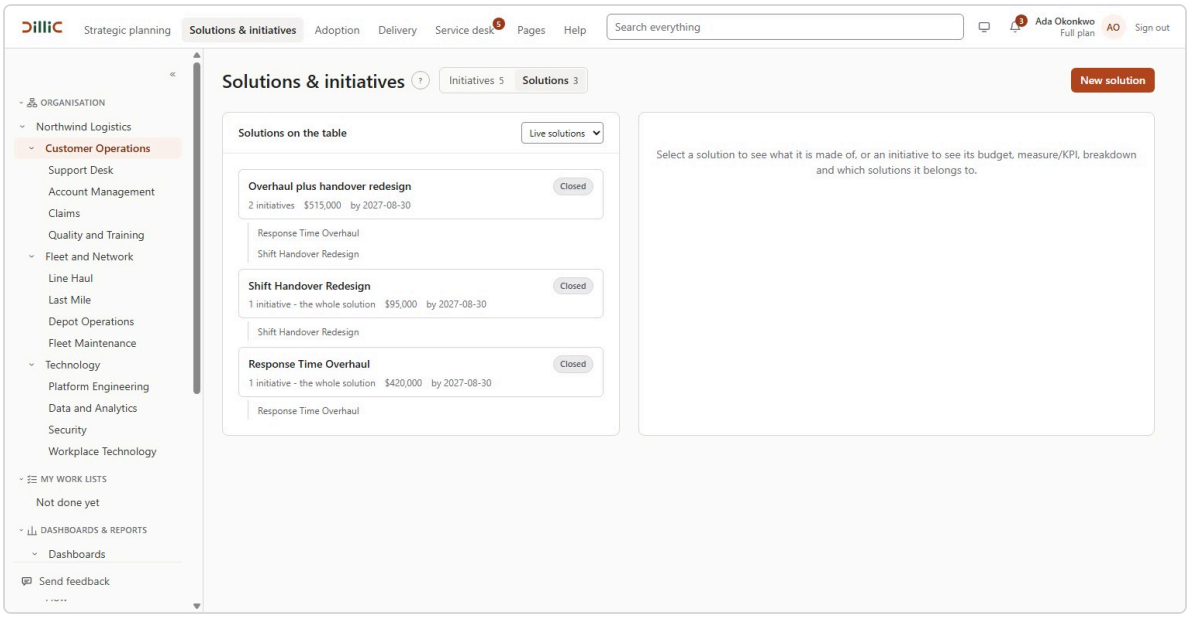
KPI readings and progress

An initiative measured by a KPI shows a bar towards its target (reversed when lower is better). Record readings under **Readings** the same way as for a factor - or, where the KPI's readings come from the services, read them: one a month, off the services the initiative establishes or enhances, so an initiative to enhance payments is judged by payments' own figures moving. If your organisation orders initiatives by a score (RICE, ICE, or WSJF; chapter 16), **Score it** sets the score's parts; an initiative missing a part stays unscored.

Breaking an initiative into work

Decomposition is the initiative's tree of work items. **Add epic** (or whatever your organisation calls the top rung) creates new work under it; **Attach existing** brings in work that already exists. The initiative also lists its risks and the change requests **releasing** it.

Solutions



Three competing solutions, each opening to its initiatives.

New solution asks which initiatives it is made of. With one, the solution is that initiative; with two or more, give it a name and optional description. **Propose** adds it to the table. Proposing does not assess it.

A solution shows how many initiatives it holds, their combined budget, and when they are assumed delivered. Its name and description can be edited; its initiatives are fixed once it has been assessed, and an assessed solution is archived rather than deleted.

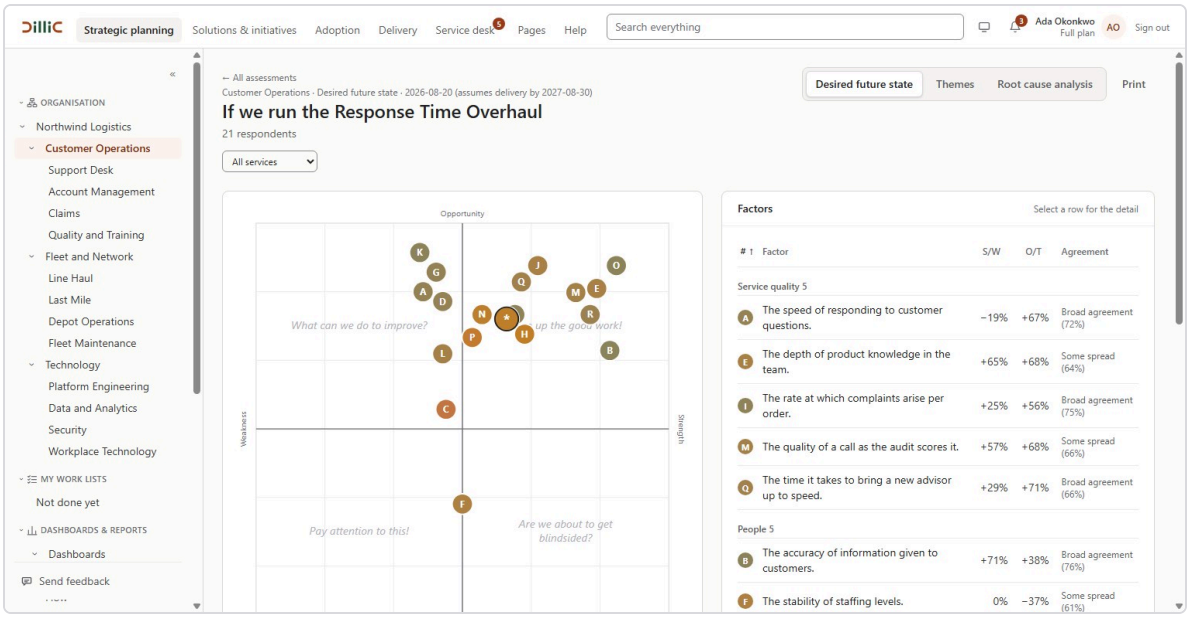
Assessing a solution

A proposed solution offers **Conduct the assessment**: a desired future state asking the same factors as though its initiatives had been delivered.

1. Accept or change the **Title** ("If we run …") and the **Assessment date**.

2. Say when it would be **Implemented by**.
3. Choose the current state it is **Measured against**, and whether answers are anonymous.
4. **Assess this solution**. Then launch it like any assessment.

Once assessed, a solution links to its assessment, to **Compare with the current state**, and to compare with rival solutions measured against the same baseline.



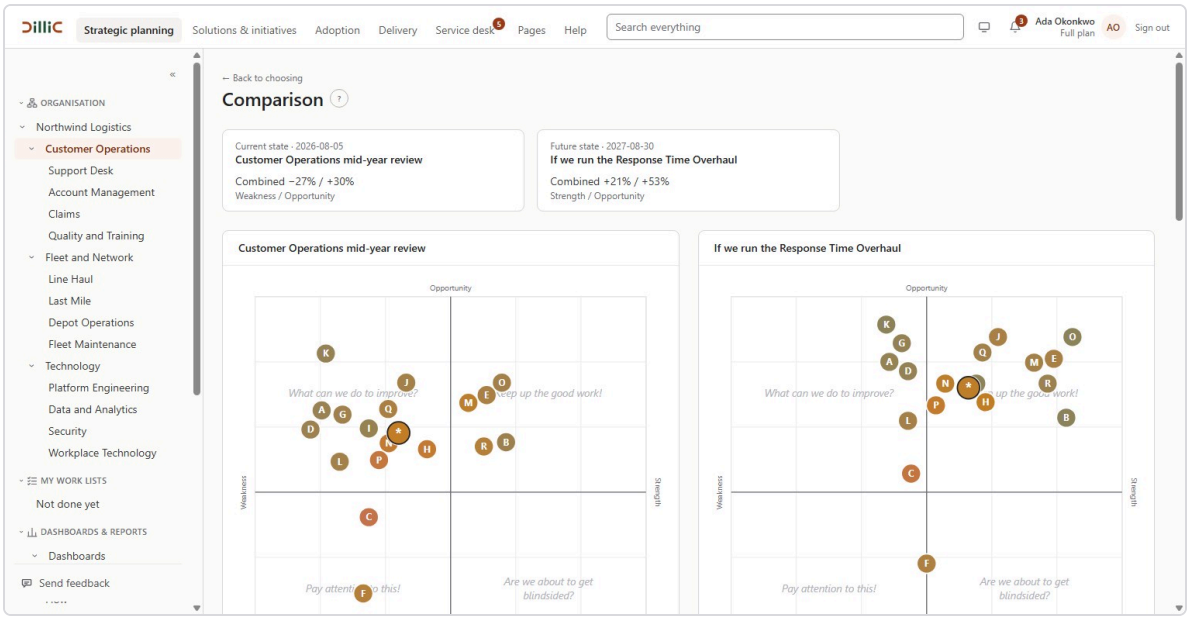
A desired future state, measured against the current state.

Comparing assessments

On **Assessments**, click **Compare**, tick two or more, and **Compare** them (or follow a solution's compare links). It has two uses:

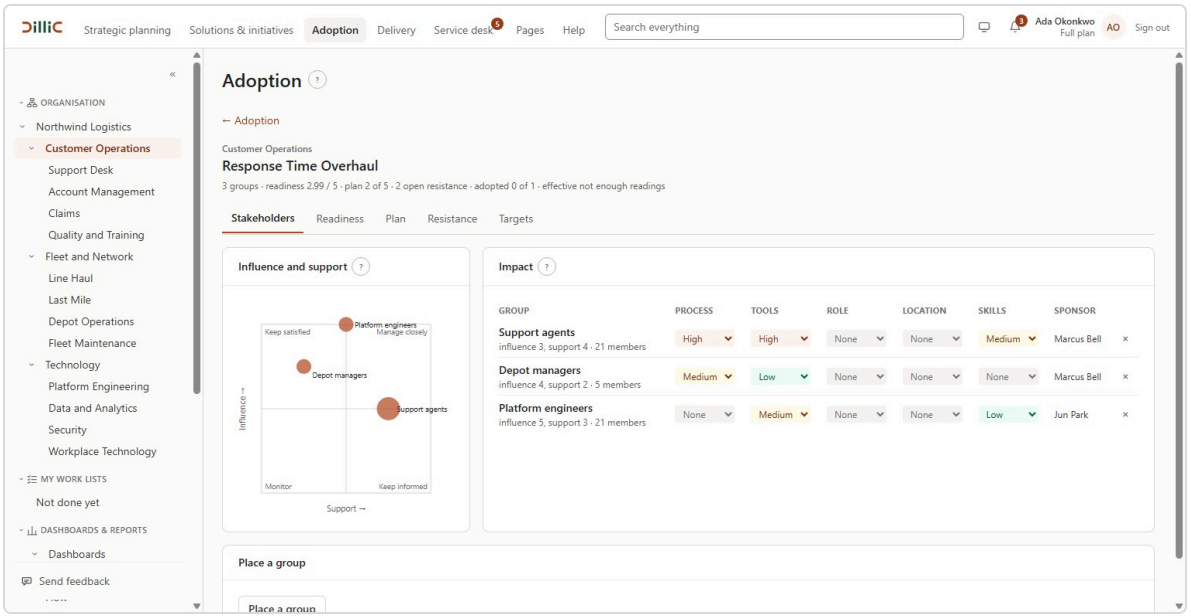
- **Choosing**: compare the desired future states of competing solutions.
- **Checking**: compare a current state with a later one to see whether anything moved.

Each assessment gets a tile and its own matrix. **Factor by factor** lines the marks up by letter, with each assessment's scores and a **Movement** column from the first to the last, in points: green where it rose by more than one, red where it fell. **KPI readings over the same span** shows each factor's KPI over the same window, so a movement in opinion can be read against a movement in the numbers. ← **Back to choosing** returns with your ticks still in place.



Compare: the current state beside a solution's desired future.

Adoption: the people the change happens to



Stakeholder groups on the grid, with the impact the initiative has on each.

Planning ends with a desired future state, and Delivery builds the work that gets there. **Adoption** in the header tracks the people between the two, initiative by initiative. The list shows each initiative's change in one line - groups, readiness, the plan's progress, open resistance, how many adoption targets are met, and whether the initiative is effective - and opens one into five tabs. Every initiative links here from its own card (**Adoption** beside **Page**).

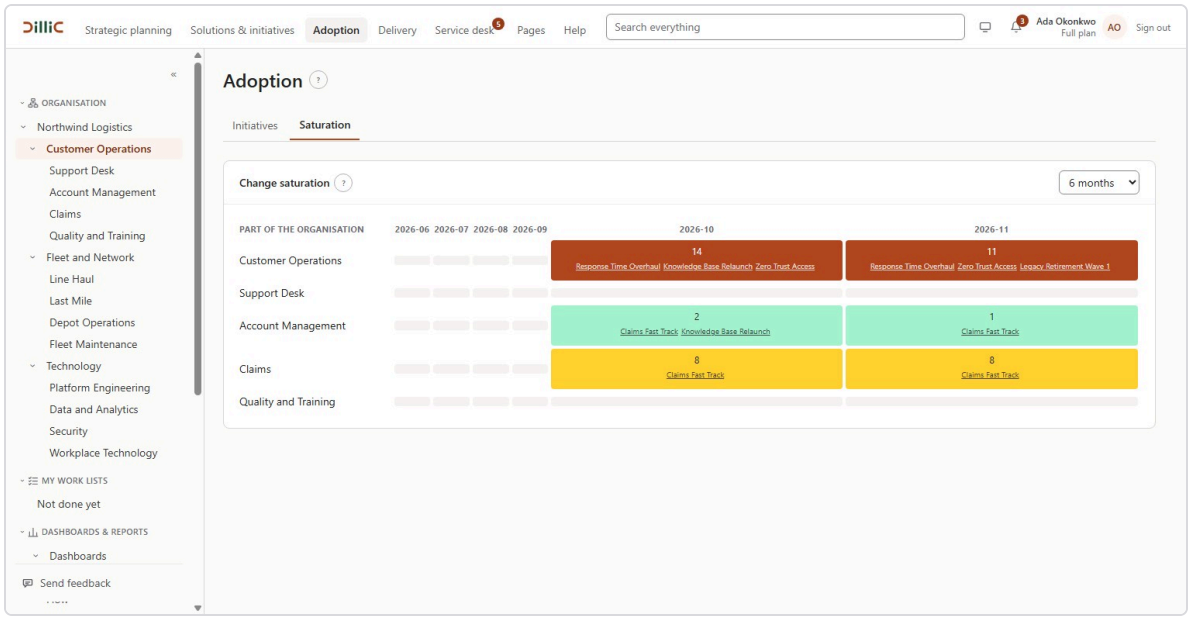
1. **Stakeholders**. **Place a group** - one of the organisation's **Groups** (Administration → Groups, chapter 14), with how many are in it; **Or name a new group** adds one to the library where you may keep groups -

with its **Influence** and **Support** (1 to 5, the classic grid), the **impact** the initiative has on it (process, tools, role, location, skills, each none to high), and a **Sponsor**. The circle's size on the grid is how much the change asks of them.

2. **Readiness**. **Open a round** for a group: a short survey of the five readiness dimensions (awareness, desire, knowledge, ability, reinforcement). By default each member of the group gets their own link by email, in their own language (**Send each member their own link**); people outside the organisation are written to as well, and anyone who unsubscribed or is inactive is left out and counted. The round shows how many have answered, and **Who was invited** lists them, answered or waiting, for those who manage adoption. **Remind** writes again to everyone still waiting, at most once a day, with a fresh link; **Invite anyone new** reaches people who joined the group since; **Withdraw** stops one person's link before they answer. For a group with nobody listed, **One shared link** works as before: the link is shown once, to pass on yourself. Either way the answers are anonymous - DilliC knows who has answered, never which answers are theirs - and scores show once three people have answered. The round reads back as an average per dimension and overall; rounds over time are the readiness curve. **Close the round** stops every link. Because results update as answers arrive, somebody watching one person turn from waiting to answered could guess what they said in a small group; read rounds once they close where that matters. A readiness letter carries the only copy of its link, so it is sent whatever somebody's notification settings say.
3. **Plan**. The communications and training plan: **Plan an activity** (communication, training, coaching, or event) with a channel, a date, an owner, and the groups it is aimed at; mark it **Done** with how many it reached.
4. **Resistance**. **Log what was heard**, from which group, why, and who owns it; record what was done about it and move it to addressed and closed.
5. **Targets**. **Add a target** - the number that says whether they changed. Choose a measure/KPI from the library (its unit, range, and direction come with it), the target value, and by when. Readings are recorded and voided the same way as a factor's, and each target shows its trend: reached, on track, off track, or not enough readings.

Saturation, on the list, is the heatmap of change landing on each part of the organisation month by month: for each live initiative touching a group there, its impact on that group, summed, in the months its activities are planned (or this month, for an initiative with no plan yet). A hot cell names the initiatives behind it.

Adopted and effective. An initiative succeeds only when two measurable things are true: the people it happens to have changed, and the change produced the result it was for. The adoption targets are the first; the initiative's own measure/KPI, set under **Measured by**, is the second. Full adoption of a change that does not work is a failure, and so is a change that would work if anybody used it. So the list, the initiative's tabs and its card on Solutions & initiatives show both halves side by side: *adopted 2 of 3 · effective: on track*. The Benefits tab on Strategic planning carries both halves up to the objectives each initiative serves.



Saturation: how much change lands on each part of the organisation, month by month.

Rights: *See how the people a change happens to are doing* reads it all; *Run the change for the people it happens to* writes. Both are the Adoption module's (chapter 12), which every organisation starts with on.

CHAPTER 9

Delivery

Delivery is where initiatives become work: boards, backlogs, sprints, releases, change requests, and requests from outside. Each team (an entity with nothing beneath it) has a board of its own; entities higher up read their teams' work rolled together.

The tabs

TAB	WHAT IT IS FOR
Board	The work in columns: move it on, group it, filter it, run the stand-up.
Backlog	Everything not yet in a sprint, in priority order; where new work is created.
Timeline	Epics and what is beneath them against dates.
Roadmap	What is planned, by quarter.
Terminal	Your own items, ready to open in an AI coding assistant.
Sprints	Only on boards that use sprints: plan, start, and complete them, with burndown and velocity.
Planning poker	Estimate together.
Risks & dependencies	What might go wrong, and what waits on another team.
Release path	Versions, change requests, and deployments in one place.
Calendar	Six weeks of change windows as bars among the sprints, freezes and days off shaded behind them, collisions warned, and who is away.
Services	What the organisation runs, so a change can say what it touches.
Planning intervals	Where the organisation runs SAFe: the train's shared cadence, planned beside the boards it writes dates onto (chapter 10).
Pages	Documents kept with the team's work.

Every item has an address of its own (`/items/…`) that opens it full-page, and you can share a list or a change request by copying the page's address.

The board

A board runs as **Kanban** (a continuous flow with limits on each column), **Scrum** (work planned into fixed sprints), or **Scrumban** (Kanban's flow and limits, planned in sprints). The badge at the top left says which.

Along the top:

- **The sprint** to show, on boards that use sprints; the running one by default. A Kanban board instead shows everything in play that has left the backlog.
- **Group by** assignee, epic, priority, type, or initiative, as horizontal lanes.
- **Search this board**: every word must match the key, title, type, status, priority, assignee, or initiative.
- **Ask**: a question in DilliC's query language, and the board's quick-filter buttons (below).
- **Stand-up**: runs the daily meeting person by person.
- **Edit**: the shown sprint's dates, goal, and capacity (for those who manage sprints).
- **Configure**: the board's methodology, columns, cards, quick filters, and statuses (for those allowed).
- Pages have left this screen for a tab of their own in the header.

When a search or question narrows the board, a line says how many cards match; column counts and their limits still count everything.

Cards

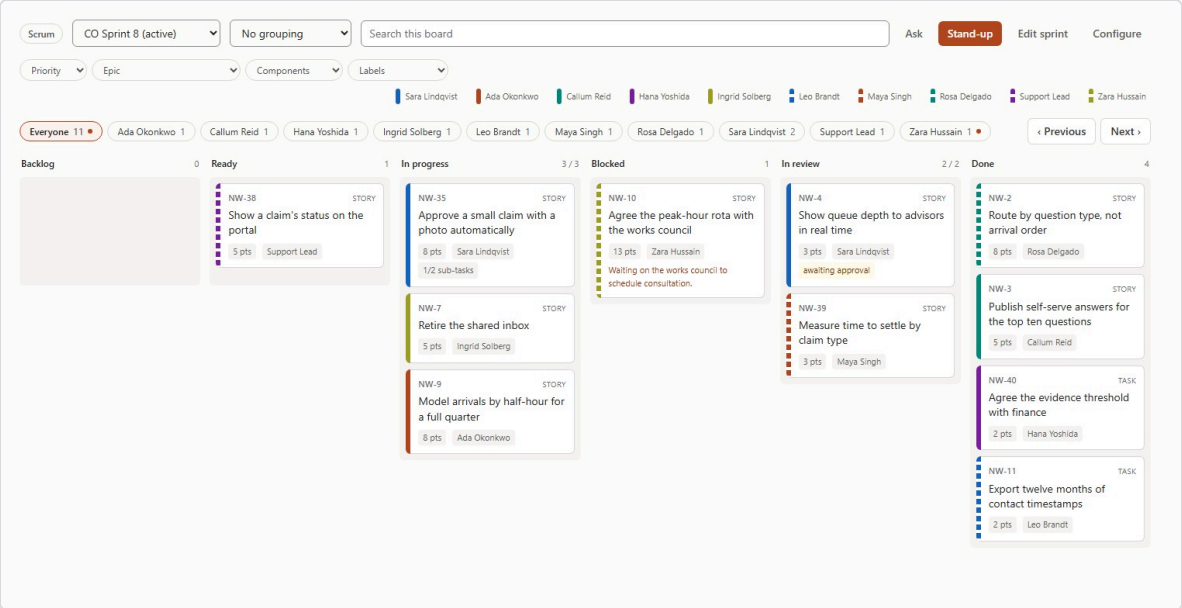
A card shows its key, type, and title, and the fields the board chose (by default the estimate and assignee). It also shows, when they apply, how many sub-tasks are done, that it is *awaiting approval*, and why it is blocked, in red. The left edge is coloured by how far the work has got, or by a field the board chose, with a legend above. The edge uses five colours; past five values they repeat dashed and then dotted, and the edge widens so the pattern shows. A card in several components or labels has its edge split, one mark for each. Click a card, or press Enter on it, to open it.

Moving work

Drag a card to another column, or use the status menu in the item (below). DilliC offers only the moves the workflow allows: columns you cannot reach are dimmed while you drag, and hovering a target says where the card would go. A column holding several statuses shows each as its own drop zone.

- **Work-in-progress limits.** A column at its limit refuses more work until something leaves. The header shows "count / limit", in red when over. The limit counts the cards the board draws: on a board that runs sprints, the cards of the sprint in progress. Sub-tasks and work outside that sprint do not count. Moves between statuses in the same column are never blocked.
- **Approval gates.** A move behind a gate does not happen: it asks for approval, the card stays put and says *awaiting approval*, and a notice explains. In the demo, calling a story done needs a sign-off from someone other than its author.
- Moving out of a blocked status clears the blocked reason. Everyone watching the item is told.

Stand-up



Stand-up: one person's cards at a time, with Previous and Next.

Stand-up puts a row of people across the top: Everyone, then each person holding work, then Unassigned. Pick one and the board shows only their cards, which stay movable. ‹ **Previous** and **Next** › walk the room in order, so whoever runs the meeting presses one key and the screen follows the conversation. While it is on, the meeting has the whole window: the header, the left pane and the Delivery heading and tabs step aside, and the **Stand-up** button stays lit. Press it again and everything comes back. It shares the row with **Ask** and **Edit sprint**, and only one of the three is open at a time: pressing another closes the one that was open.

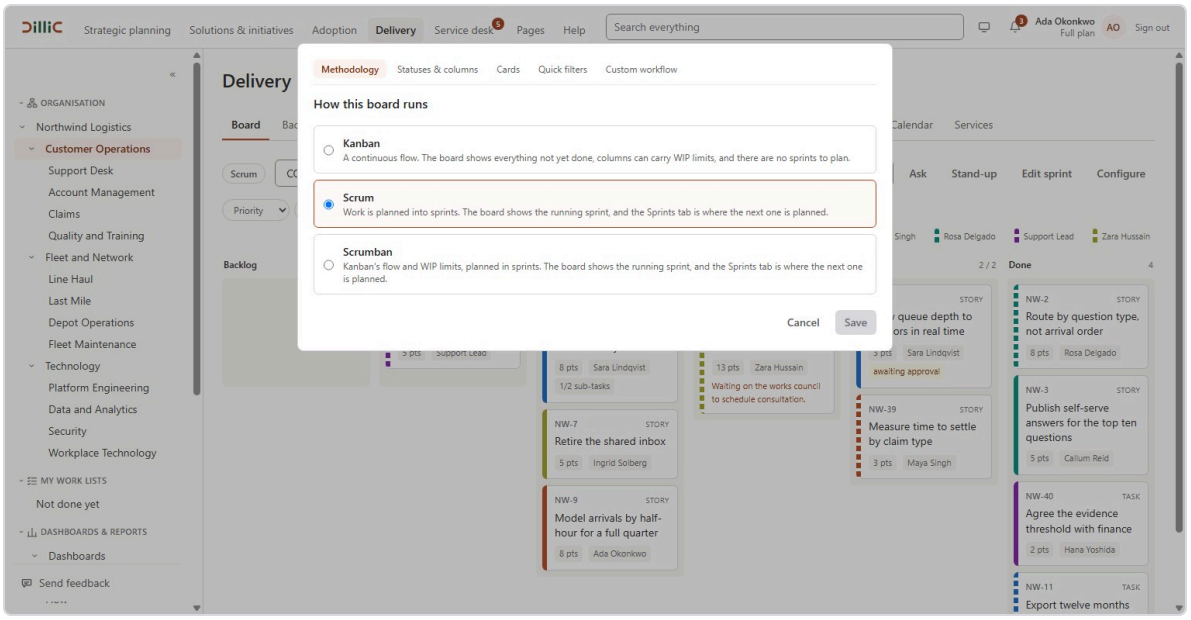
Asking questions of the work

Ask opens a line where you type a question such as `priority is Critical and assignee is me`. As you type, DilliC says what it understood ("Reads as …") or what it could not read. Press Enter: the board keeps its shape and hides the cards that do not answer. The same language works in the backlog, in lists, in reports, and in automation; appendix A describes it in full.

Quick filters are buttons a board keeps for questions it asks every day ("Mine", "Blocked"). They appear in the Ask row; press several and a card must answer all of them. **Clear** releases them. Hover one to see its question.

Configuring a board

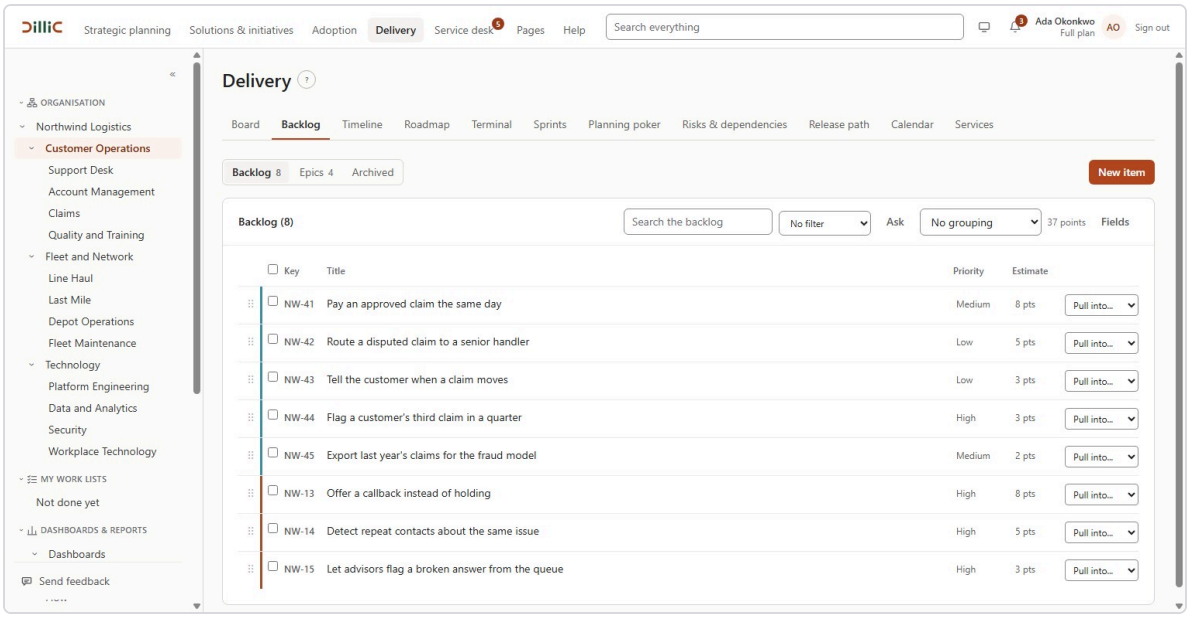
Configure opens five panes. Each saves on its own; switching pane abandons unsaved edits.



Configure → Statuses & columns: which statuses make up each column, and its limit.

Methodology	Kanban, Scrum, or Scrumban. Switching to Kanban removes the Sprints tab, and is refused while a sprint is running. SAFe is offered only on a board already running it.
Statuses & columns	Each column's name, the statuses it holds, its work-in-progress limit (blank for none), and its order; **Add a column**; and the statuses available but not on the board. Every status holding work, and the status new work starts in, must be on the board, and no status can be in two columns. Where the organisation runs several workflows, choose this board's here: DilliC first shows where the work would land (each card moves to the status that means the same) and then **Switch to …** moves it.
Cards	Which fields a card shows (priority, estimate, assignee, reporter, epic, due date, components, labels, score, dates, or custom fields), and what colours its edge.
Quick filters	Each button's words and its question, checked as you type. Anyone can press them; changing them needs the right to configure boards.
Custom workflow	Statuses of the board's own ("Awaiting customer"), each saying which of the seven states it is, and moves of the board's own between them. Offered only where the organisation allows it (chapter 17). A new status sits in no column until you put it in one.

Backlog and new work



The backlog: what is not yet in a sprint, with Epics and Archived beside it.

The **Backlog** lists open work not in a sprint, in the team's order, a hundred at a time. **Epics** lists the containers, and **Archived** what has been put away (with **Bring it back**). Search, Ask and Group by work as on the board; clicking a column heading sorts by it. The total of points shows beside them.

- **Reorder**: drag a row by its grip (⠿), or focus the grip and use the arrow keys. The grip appears while the backlog is in its own order: not sorted by a column, searched, asked, filtered, or grouped. Only the rows you moved change places; the rest of the backlog keeps its order.
- **Edit in place**: priority, assignee, reporter, estimate, and due date change in the row. Status is changed by moving the item, not by editing a cell.
- **Pull into…** a row's menu puts it into a sprint that is not finished.
- **Bulk**: tick rows to **Move to…** a status or **Set priority…** for up to 200 at once. A bulk edit is all or nothing; moves report how many happened, how many wait on approval, and why any did not.
- **Fields** chooses the backlog's columns (for those who configure boards).

Creating work

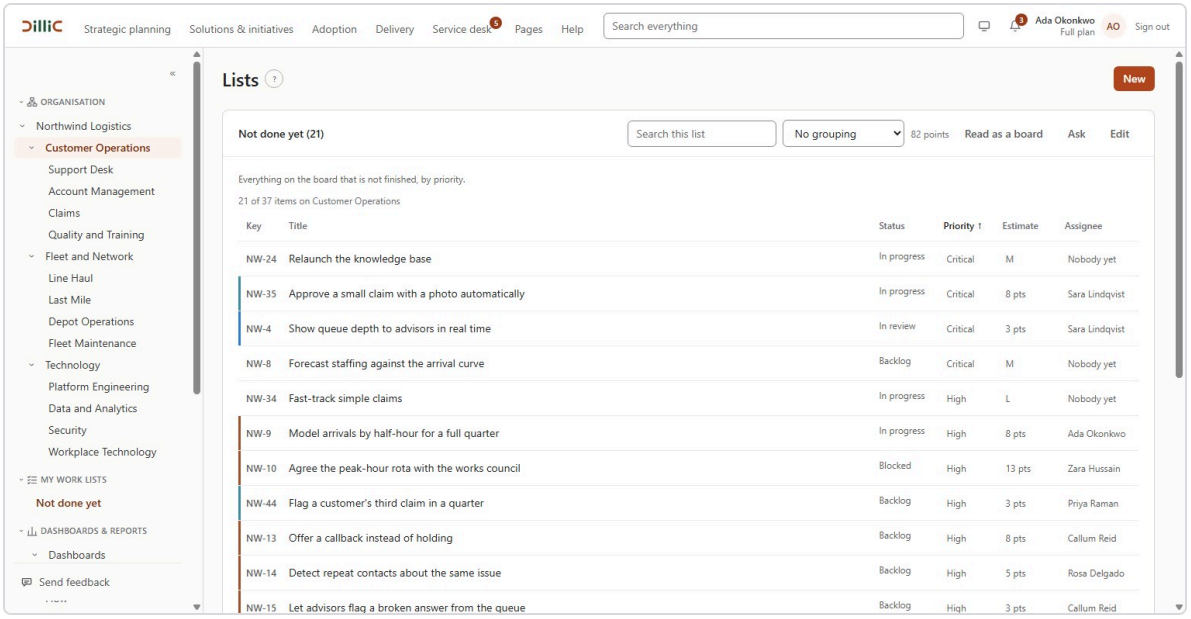
1. **New item**.
2. Choose the **Type**: by default Epic, Story, Task, or Defect.
3. Choose what it **Sits under**: an item one rung up (an epic for a story), or for an epic, an initiative. It takes its initiative from its parent. Or choose "Nothing — it sits on its own".
4. Give it an estimate: points for stories, tasks, and defects, a T-shirt size for epics.
5. Write the **Title**, and optionally a description and (for stories) acceptance criteria.
6. **Add**. Assign it from the item afterwards.

The rungs of work

RUNG	BEHAVIOUR
Epic	A container under an initiative. Sized in T-shirt sizes, never on the board and never in a sprint or version.
Story, Task, Defect	The work a board shows. Estimated in points, planned into sprints. Only stories carry acceptance criteria.
Sub-task	A line on a checklist inside its parent: ticked off rather than moved, with no estimate, assignee, or card.

A parent is always exactly one rung up. Your organisation may have renamed these rungs or added its own (a "Spike" behaving like a task, say); with SAFe on, epics read as features, and enabler features and stories appear.

Lists



A list open on the Lists screen, with New at the right of the heading.

A **list** is a question about the work, saved under a name and asked afresh every time it is opened, so it always shows what is true now. Lists are not a Delivery tab: they are kept in the left pane under **My work lists**, and each opens on the **Lists** screen. A list is yours alone. Nobody else sees it, and it cannot be shared.

1. **New**, at the right of the Lists heading. Write the conditions: **Build it** picks field, test and value; **Type it** takes the query language. Choose "Match all" or "Match any".
2. Choose its **Data source**: the board it is written on, boards you pick, everywhere you can look, or **Changes with the selected entity**. That last one follows whatever is chosen in the Organisation tree: the

entity's own board, or, for an entity with no board of its own, the boards beneath it. A list can pull data from one or more boards you have access to.

3. Pick **Columns** and **Sorted by**.
4. Name it ("Unestimated and unassigned") and say what it is for.
5. **Create the list**. It appears under My work lists.

An open list can be searched and grouped, and read as rows or **Read as a board**. A condition naming `me` means you. A list never shows you work you could not otherwise see: if you lose access to a board it asks of, the answer leaves that board out.

Sprints

The **Sprints** tab lists each sprint with its dates, points committed, done and carried over, and its state. **Plan a sprint** asks for a name, start, length (the end follows), capacity in points, and a goal. Once the sprint is planned, **Who can take what** sets capacity per person; the sprint's capacity is then their sum.

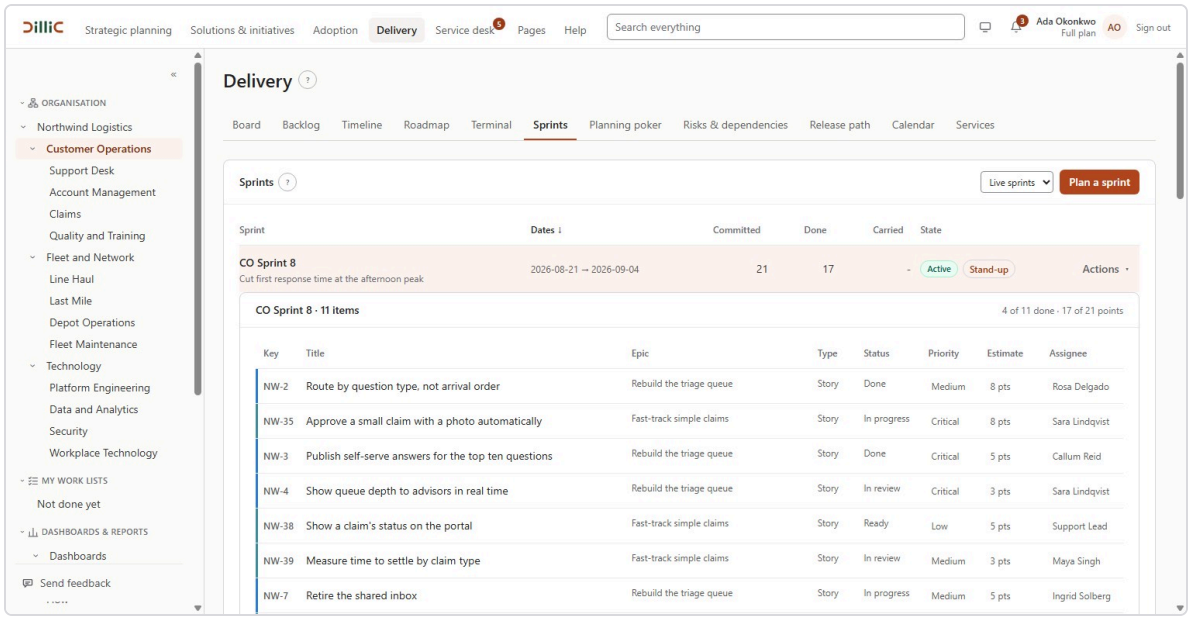
While planning, **Who has what** on a sprint's menu adds up the points in it per person, with what nobody has picked up as a row of its own and the total beneath. Items with no estimate are counted beside the table rather than as zero, and capacity shows only where somebody has set one. Anyone who can see the board can open it.

Start	Only one sprint runs at a time. Starting records what the team committed to.
Complete	Records what was done and what carried over, and returns every unfinished item to the backlog, noting it in each item's history. A completed sprint's dates are fixed from then on; its name, goal, and retrospective can still change.
Archive	For completed sprints only; **Bring it back** restores one.

Each sprint's menu opens:

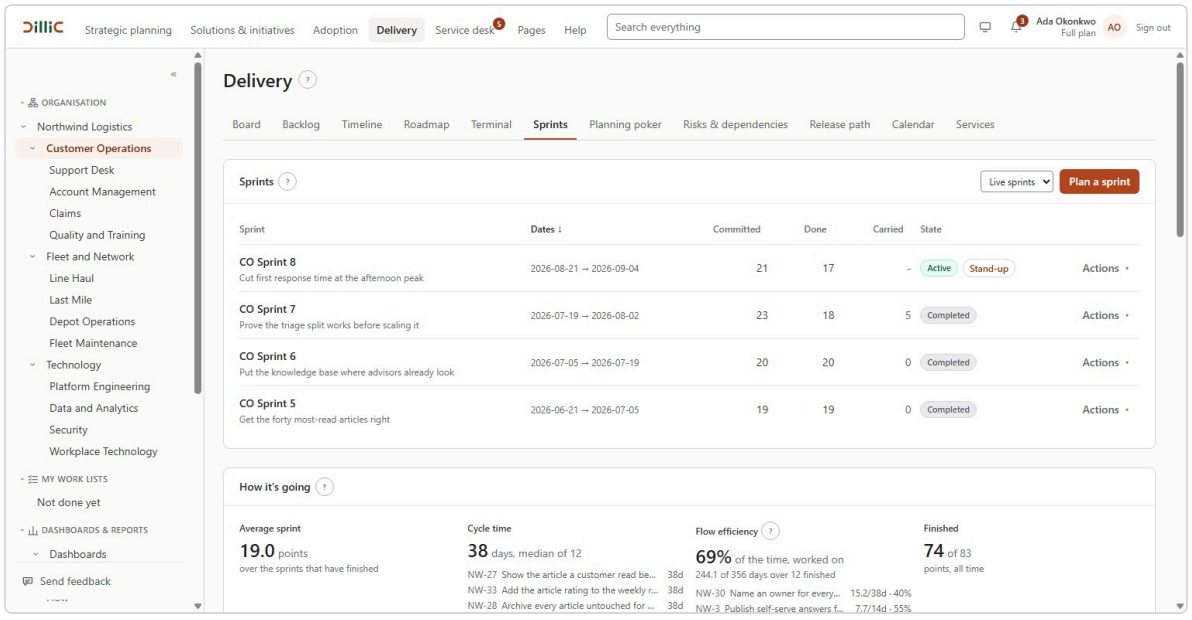
- **Items in this sprint**.
- **Stand-up**: per person, blocked first, then in progress, not started, and done, with anything done since yesterday marked. It has the whole window while it is open; the lit button on the sprint's row closes it.
- **Burndown**: points left against the ideal line. **Burnup**: points done against the scope.
- **Report**: what finished, what did not, and what was added or taken out after it started.
- **Retrospective**: a write-up, which the AI assistant can draft from the sprint's numbers.

DilliC draws charts from what happened, move by move, so re-estimating a story today does not rewrite last sprint's chart.



A sprint opened to its items.

How it's going, below the sprints, gives the average sprint, cycle time, flow efficiency, points finished per sprint for the last eight, what recent time went into, and a control chart and cumulative flow over dates you choose. Velocity counts work completed, never cancelled. **Flow efficiency** is the share of the cycle time spent in progress or in review rather than waiting: ten days from start to finish with two of them worked on reads 20%, which is a queue to clear, not a team to hurry. The sprint report gives the same figure for one sprint.

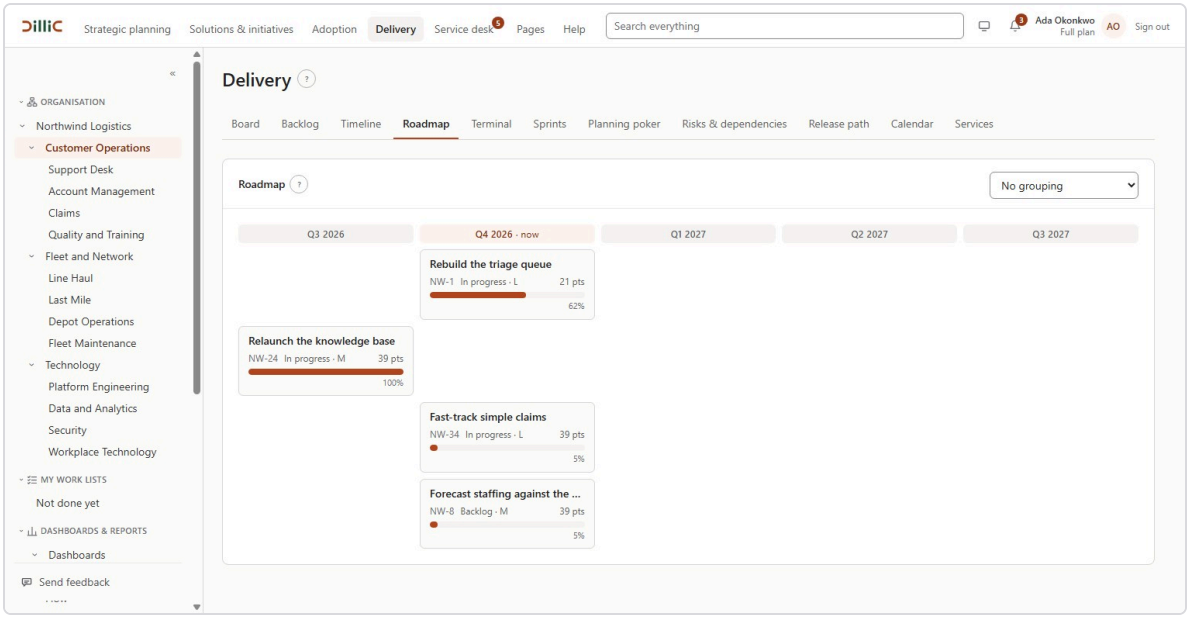


Sprints, and how it is going: the average sprint, cycle time, and what has finished.

Timeline and roadmap

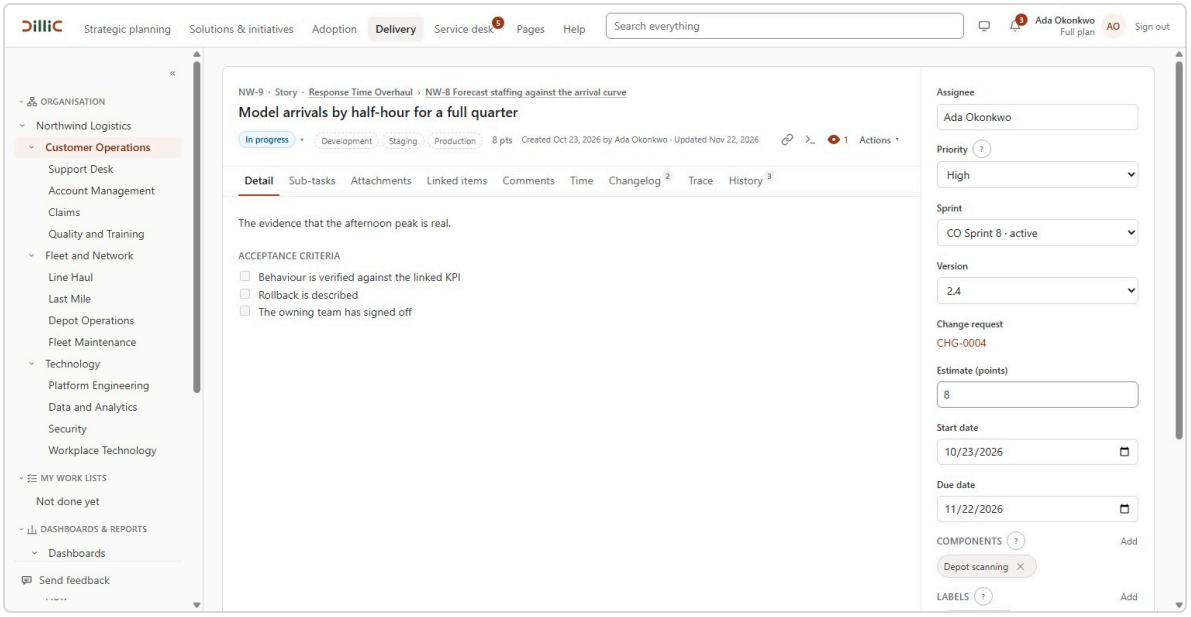
Timeline draws epics as bars against dates, zoomed by days, weeks, or months, with **Today** to jump back. A solid bar uses the item's own dates; a dashed one takes them from the work beneath; the fill shows how

much beneath is done. Open a row to see its stories. **Roadmap** lays the same things out by quarter, with anything undated listed as "Not yet dated", grouped by initiative, priority, or status. Both read from the work itself, so they are never a separate plan to keep up to date.



The roadmap for Customer Operations.

A work item



A work item: its trail, status, tabs, and the fields beside it.

The top of an item shows its trail (reference, type, initiative, and parents; click one to go there), its title (click to edit), and its **status**. The status menu lists the moves you can make: back a step, into blocked, on a step, cancel, and any other move the workflow allows. A gated move asks for a note to the approvers, then **Ask**

for approval. Beside it you can copy the item's link, copy the command to work on it in a terminal, open it in its own tab, and **Watch**. **Actions** offers:

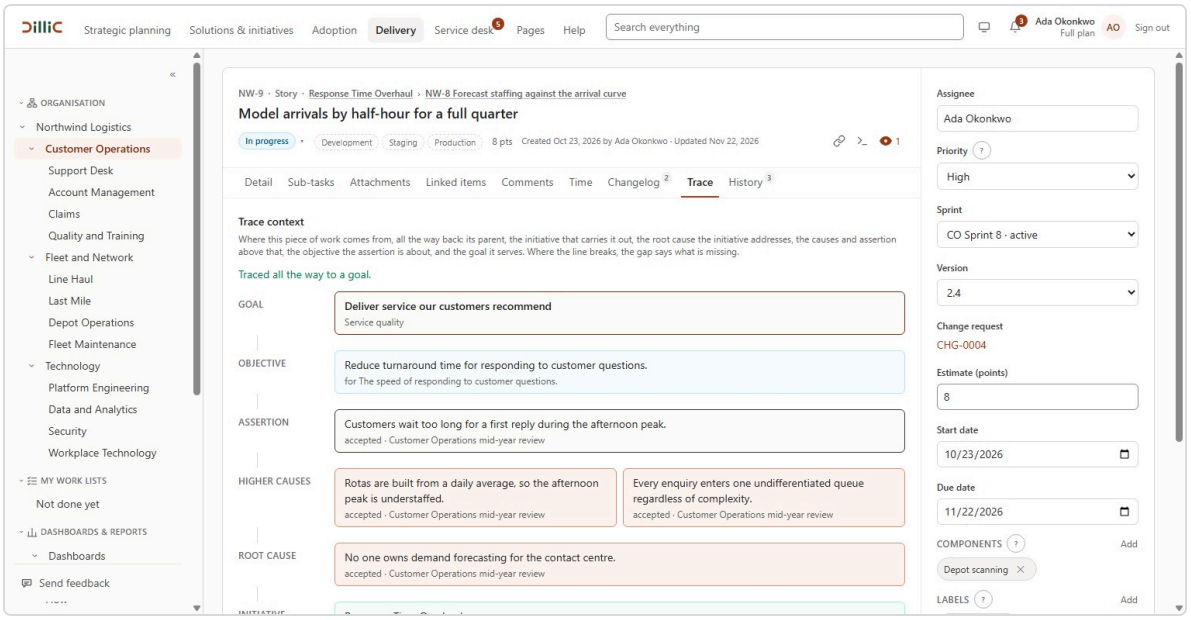
- **Copy**: the same wording, and what is beneath it.
- **Move**: to another part of the organisation; everything beneath travels with it.
- **Archive**: once the item is released or cancelled, or for a container, once everything beneath is. Archived work is kept and searchable.

The fields beside it:

- **Assignee** and **Priority** (or, where the rung is scored, its score with **Score it**).
- **Sprint**, and **Version** (an unreleased one). **Change request** is read from the version: which change request, if any, authorises it.
- **Estimate**, **Start date**, and **Due date**.
- **Blocked reason**, while it is in a blocked status: what is it waiting on?
- **Components** (from the organisation's list; "None fit — ask for one" asks an administrator), **Labels** (type a new one to create it), and the organisation's own **Fields**, required ones marked *.
- **Sits under**, with **Change** to move it under something else.

The tabs:

Detail	Description and acceptance criteria. Where environments are set up, a trail of where the item has been deployed ("is my fix out yet").
Sub-tasks	A checklist to tick off and reorder, or, for other rungs, the work beneath, with **Add child** to create new or adopt existing.
Attachments	Files up to 10 MB each.
Linked items	Sideways links, each with a reason read from both ends: blocks / is blocked by, duplicates, causes, relates to. A link shows at both items and can cross teams.
Comments	A thread two levels deep. `@` mentions someone, `[[` links a page, `#` links a work item, and a pasted picture goes in. You can reword your own; removing one leaves its replies.
Time	Original estimate, time spent, and time left. **Log time** takes how long (`1h 30m`, `2d`), the day, and a note, and either takes it off what is left, sets what is left, or leaves it.
Changelog	What was done, a line at a time, with commits and pull requests among them. Lines are logged once the work has started.
Trace	Where the work comes from, all the way back: parent, initiative, the root cause it addresses, the causes and assertion above, the objective, and the goal. Where the line breaks, it says what is missing.
History	Every change to the item, who made it and when.

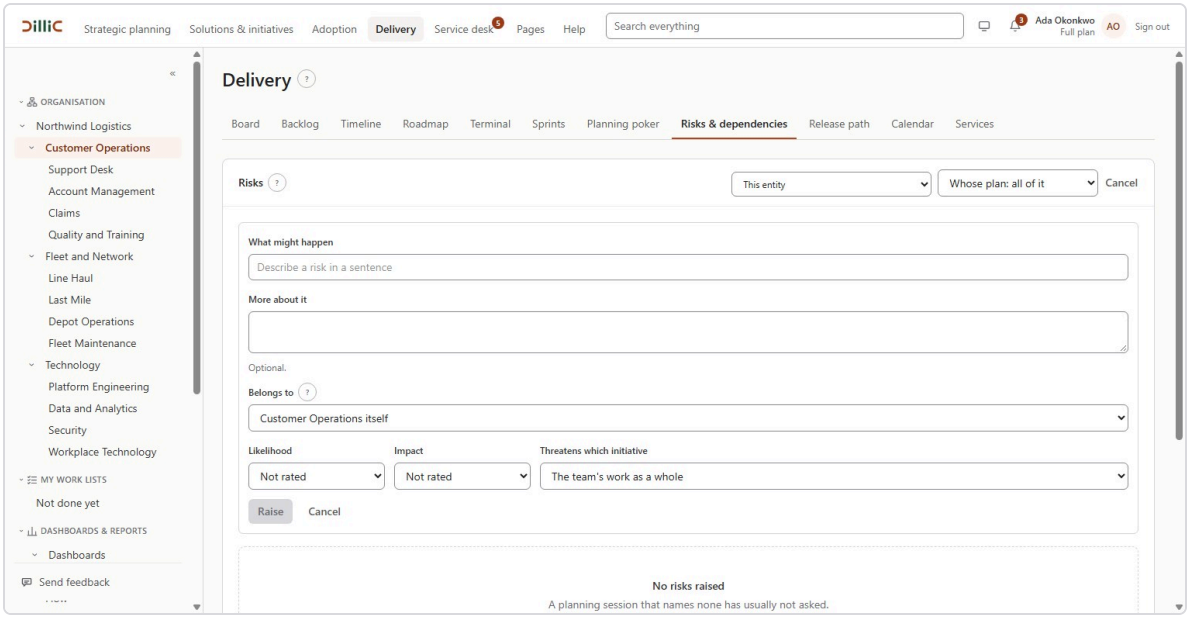


Trace: from a story back through its initiative, root cause, and objective to the goal it serves.

Approvals on work

When a move is waiting on a gate, the item shows an amber band above its tabs to the people who may decide, with a comment box, **Approve**, and **Reject**. Each person decides once. The move happens when enough people have approved; a single rejection ends the request. A gate may bar the item's author from approving, and may name the role whose holders decide.

Risks and dependencies



Raising a risk.

Raise a risk with **What might happen**, more about it, its **Likelihood** (rare to almost certain) and **Impact** (negligible to severe), and which initiative it threatens. Then answer it the ROAM way:

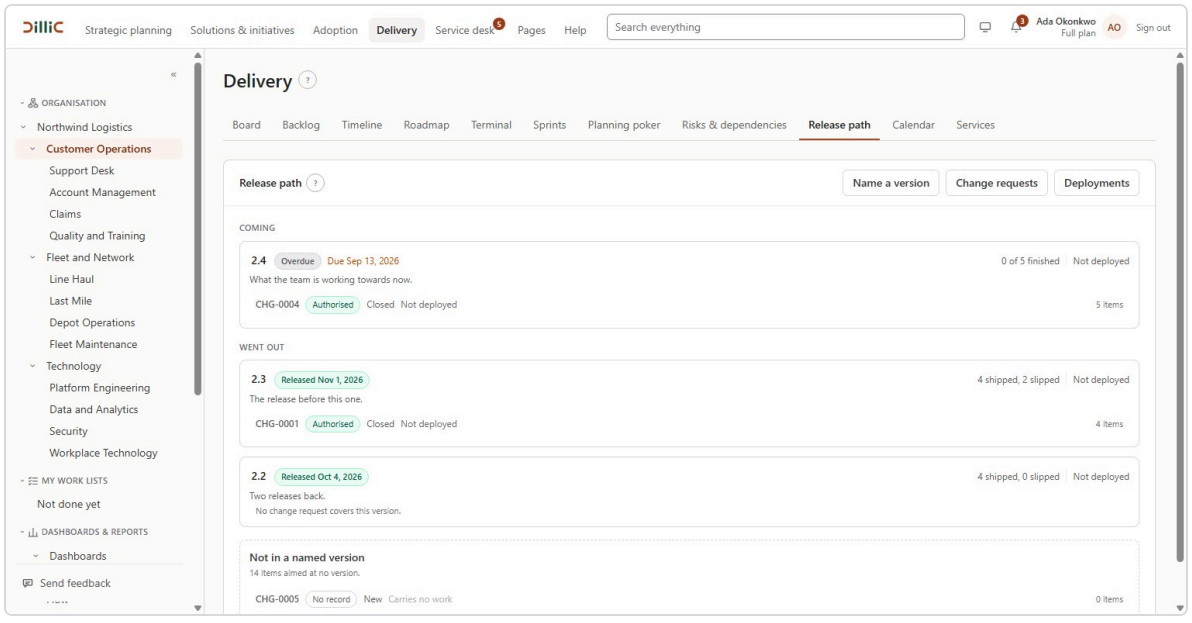
ANSWER	MEANING
Open	Raised, and not yet answered.
Resolved	Looked at, and it turns out not to be a risk.
Owned	Still a risk; somebody is named and will carry it. Needs an owner.
Accepted	Real, and nothing can be done.
Mitigated	There is a plan. Say what is being done.

Show the matrix places risks on likelihood × impact: up to 4 is low, 9 moderate, 16 high, and above that extreme.

Scope reads the register for this entity alone or for **This entity and everything beneath it**. Read for a division, it lists every team's risks beneath it that you may see, the worst rated first, each saying which part of the organisation it belongs to, and a line above the list counts them, the parts they come from, and those rated high or extreme. A risk is still raised and answered on its own team's register. The choice is remembered on this device.

Dependencies are read from links that cross teams, in two lists: work waiting on another team, and another team waiting on us. A dependency is flagged late when the work needed arrives no earlier than the work that needs it (judged by sprint end or due date), and late ones come first.

Versions and releases



The release path: versions coming and gone, their change requests and deployments.

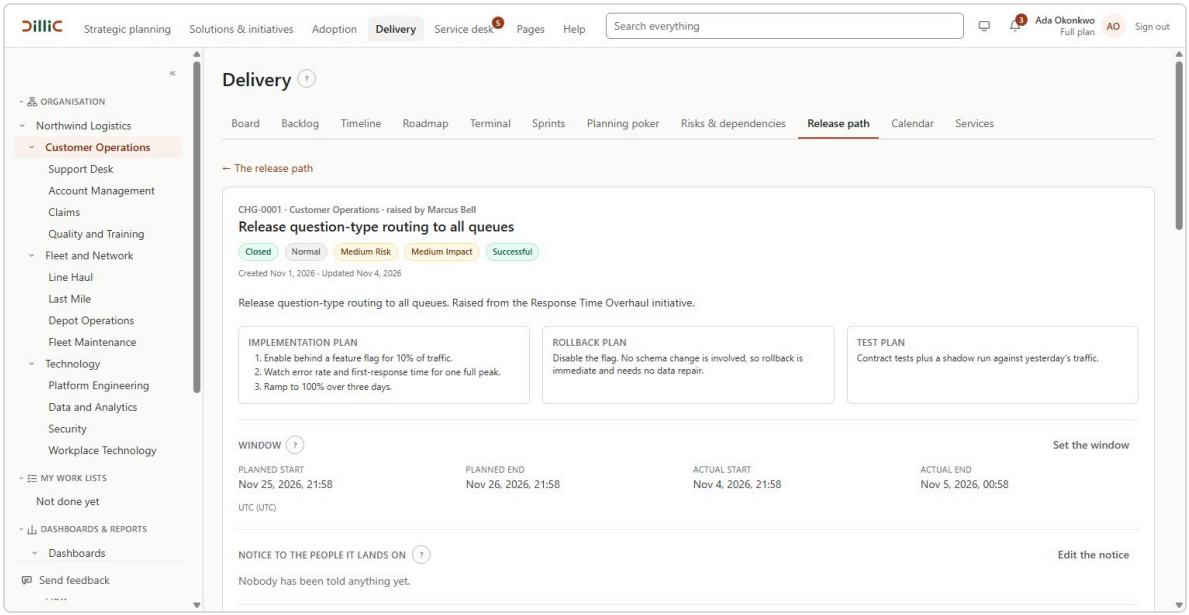
Release path lists versions **Coming** and **Went out**. Each shows its state (unreleased, overdue, or released), when it is due, how much is finished or shipped and slipped, which environments it has reached, and the change requests authorising it, with a warning if work went out under none.

1. **Name a version**: a name ("2.4", "August release"), a due date, and notes.
2. Open it and **Aim work at** it: items by name, or everything under an epic. DilliC offers only work not already on another live version.
3. When it ships, **Release** it and say where unfinished work goes: to another unreleased version, or off the release. Releasing never waits for unfinished work; the version then shows what shipped, slipped, and was dropped.

A released version is archived rather than deleted. Once a change request authorising a version has been submitted, work cannot move in or out of it without withdrawing the change request first.

Change requests

A **change request** is how a change reaches production under ITIL change enablement. **Change requests** on the release path lists them for this entity and everything beneath; **Mine to approve** shows only those waiting on you.



A closed change request with its plans, window, and the version it authorised.

1. **Raise a change request**: its title, **Type** (Standard, Normal, or Emergency), **Risk** and **Impact** (low, medium, high), the implementation, rollback and test plans, and the planned window with its time zone.
2. Add the **versions it authorises**. The work it releases is read down through them.
3. Move it through its lifecycle with the buttons under **Move it**, adding a note for the change authority:

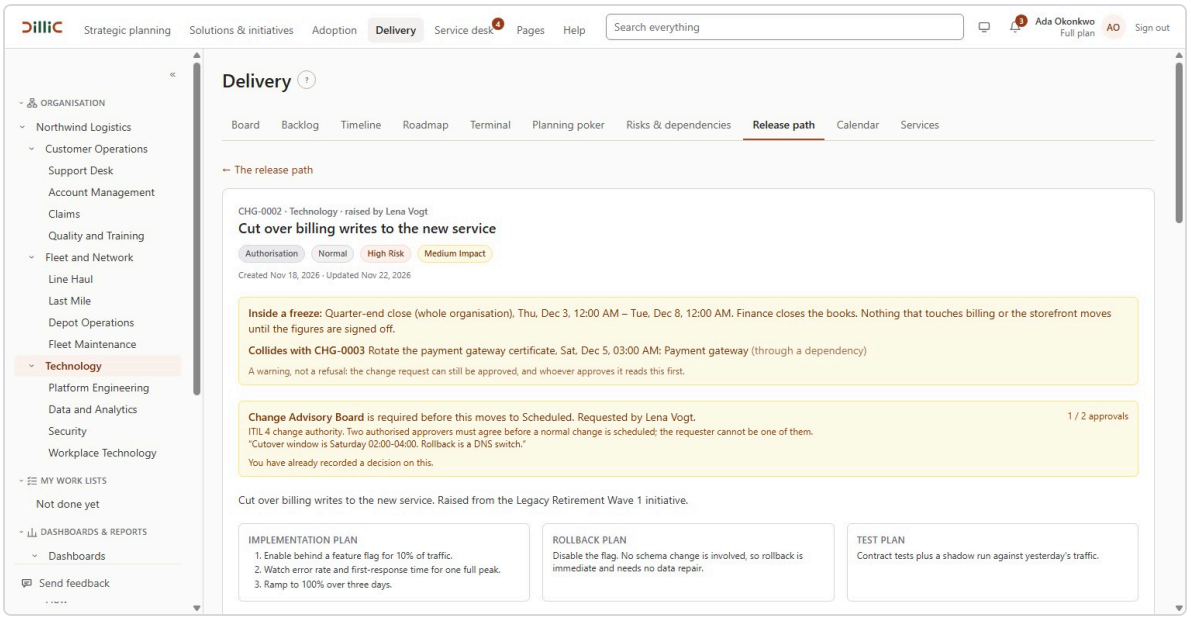
FROM	MOVE	TO
New	Assess risk and impact	Assessment
Assessment	Submit for authorisation	Authorisation
Authorisation	Authorise (Change Advisory Board: two leaders, not the requester)	Scheduled
New	Schedule (pre-authorised), standard changes only	Scheduled
Scheduled	Begin implementation	Implementation
Implementation	Implementation complete	Post-implementation review
Review	Close	Closed

A change request can be sent back for rework from authorisation, cancelled from anywhere, and reopened from cancelled (but not from closed). Your organisation may have changed this workflow (chapter 17).

- It cannot be submitted while work it releases is unfinished, or go to the change authority without both an implementation and a rollback plan.
- **Set the window** edits the planned and actual start and end. The actual times are also stamped as it enters and leaves implementation, and can be taken from what the deployments did.
- At review, **Record outcome**: successful, successful with issues, failed, or rolled back, and what actually happened. A change request archives itself two weeks after its outcome is recorded.
- **Start from a model** on the raise form fills it from one of your organisation's change models (chapter 17): the plans, the services, the notice and the type. Anything you typed first stays. The model's answers to the risk questions score the change, and the change says which model it came from.

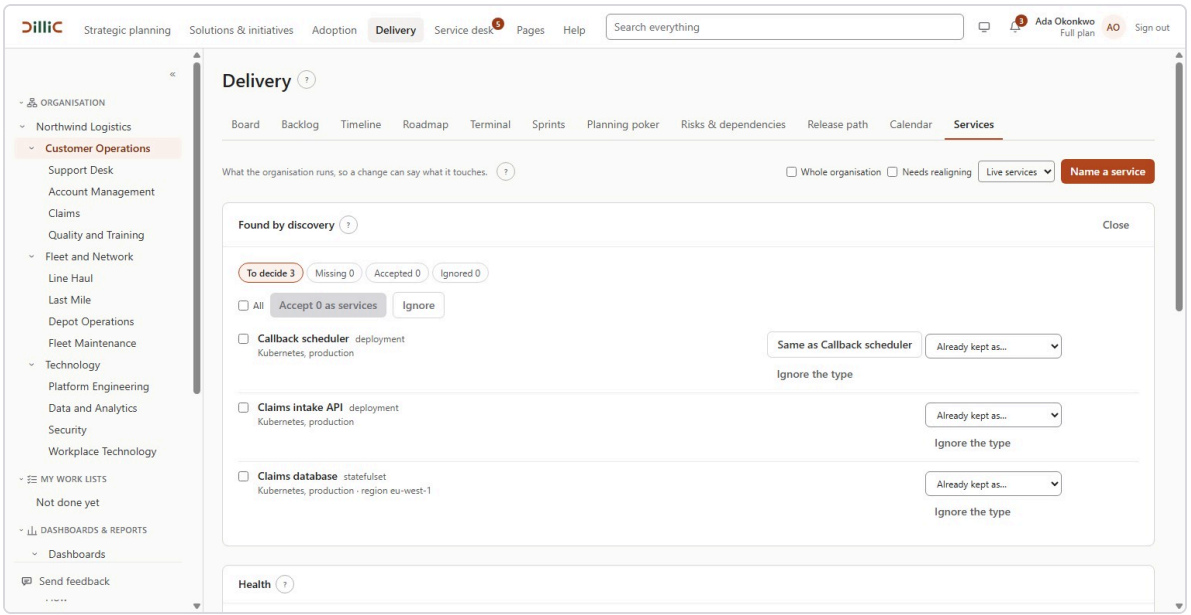
Telling the people it lands on. Under **Notice to the people it lands on**, **Edit the notice** and write what the people who use the services should know: what stops, for how long, and what to do meanwhile. Tick **Announce on the portal** and save. An announced change with a planned window is listed, with its window, services and notice, on the service desk's portal and on the page of each service it touches, until a week after its window ends. Its plans and its assessment stay inside. Announcing is recorded in the audit trail and posted to any webhook that asks for `change_request.announced`.

Assessment. Risk and impact are scored from your organisation's questionnaire (chapter 12, Risk assessment and routing). Under **Assessment** on the change, answer each question; the two levels follow the score - under a third of the possible points is low, under two thirds medium, the rest high - and the level chooses which authority the change goes through. **Set the risk by hand** overrides the score, and the change says who did and what the questionnaire had scored; setting it back clears that. **Consequences** lists the incidents the change caused and the problems it was raised to fix, and **Raise an incident** raises one against it on the desk, with the change named as the cause and its services carried over.



A change scored by the questionnaire, the services it touches, and the warning above its approval.

Services found by discovery



What a source reported: accept it as a service, join it to a service already kept, or ignore it.

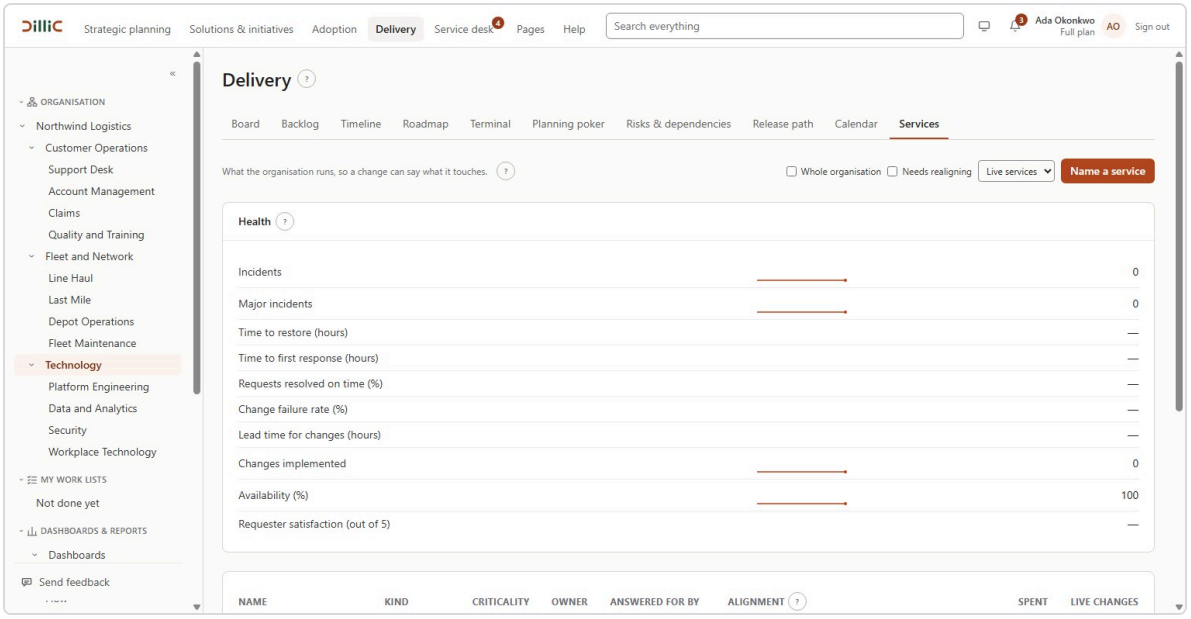
Where a cloud account, a cluster or a Terraform state already knows what you run, it can report it, so that nobody types the list. DilliC scans nothing and installs nothing on your network: a command you schedule prints the inventory and posts it.

1. **Add a source.** An account administrator opens Administration, then **Discovery in** under Connections, and adds a source: its name, the part of the organisation its services are kept on, and what it sends. The screen shows a secret once and a one-line command. Set the secret as `DILLIC_DISCOVERY_SECRET` where the command runs, and run the line on a schedule.

2. **What it sends.** AWS, Azure, Google Cloud, Kubernetes and Terraform each read what that tool's own command prints. DilliC's own shape is a list of assets, each with an `id`, a `name`, a `type`, a `kind`, `attributes` and `depends_on`. **Upload a report** takes the same JSON as a file.
3. **Decide what was found.** Services shows how many assets wait. **Review what was found** lists them. **Accept as services** makes a service of each one chosen. Where an asset is a service you already keep, choose it under **Already kept as**. **Ignore** sets an asset aside, and **Ignore the type** stops the source mentioning that type again.
4. **What follows.** A service's page says where it was found and when it was last seen. Where two accepted assets depend on each other, their services do. Where a source reports its whole inventory, an asset it stops reporting is listed under **Missing**, and its service says since when.

A report carries five thousand assets at most; filter it where it is made. Accepting needs the right to manage services where the source's services are kept.

Services and the change calendar

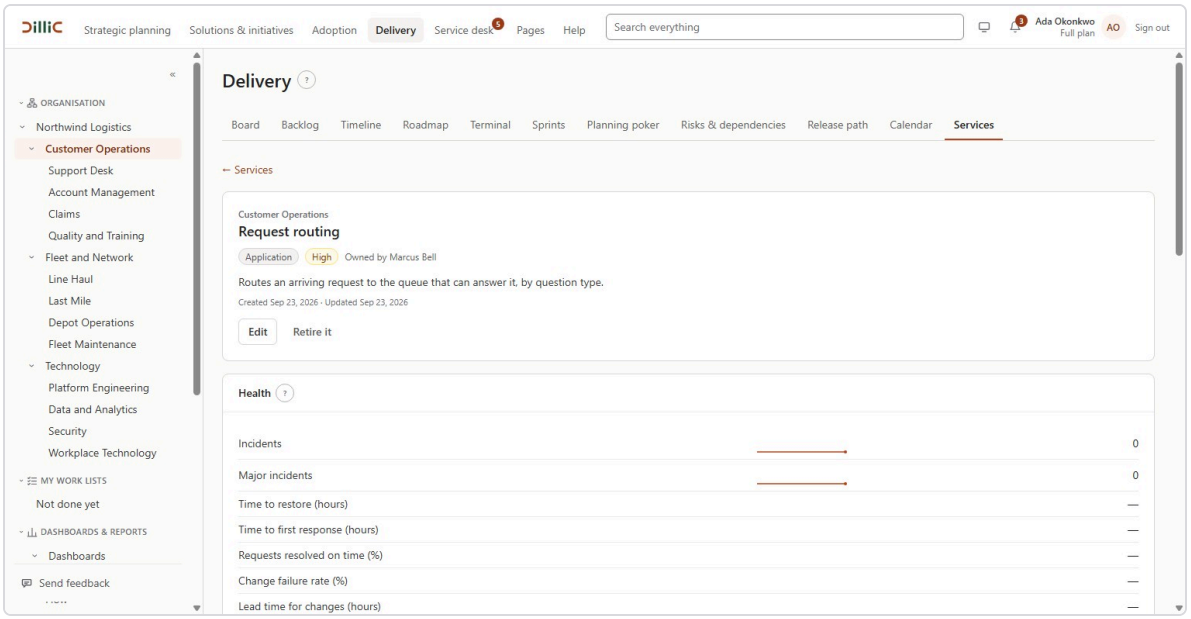


Services: what the organisation runs, with owners, criticality, and live changes.

A change that cannot say what it touches cannot be assessed, scheduled around, or blamed. **Services** on Delivery names what the organisation runs - ITIL's configuration items, under whatever noun your organisation uses (chapter 12, Vocabulary) - each with a **Kind** (application, infrastructure, platform, or business service), a **Criticality** (low to critical), an **Owner**, the entity that answers for it, and optionally what it is **Part of**. **Whole organisation** reads every entity's services rather than this one's and those beneath it.

1. **Name a service**, then open it and add what it **Depends on**. Payments depends on the database. A dependency is how a collision is read; a grouping (part of) is not.
2. On a change request, **Touches** names the services it touches - when it is raised, or during the assessment. A desk request names the service it is about the same way.

3. A service's page lists every change against it, live first, every request about it, and what depends on it.
4. A service's **Stage** is **Planned** while an initiative is still establishing it and **Live** once it runs; whoever keeps the services presses **Put it live**. **Retire it** takes a service out of the pickers. Its history stays.

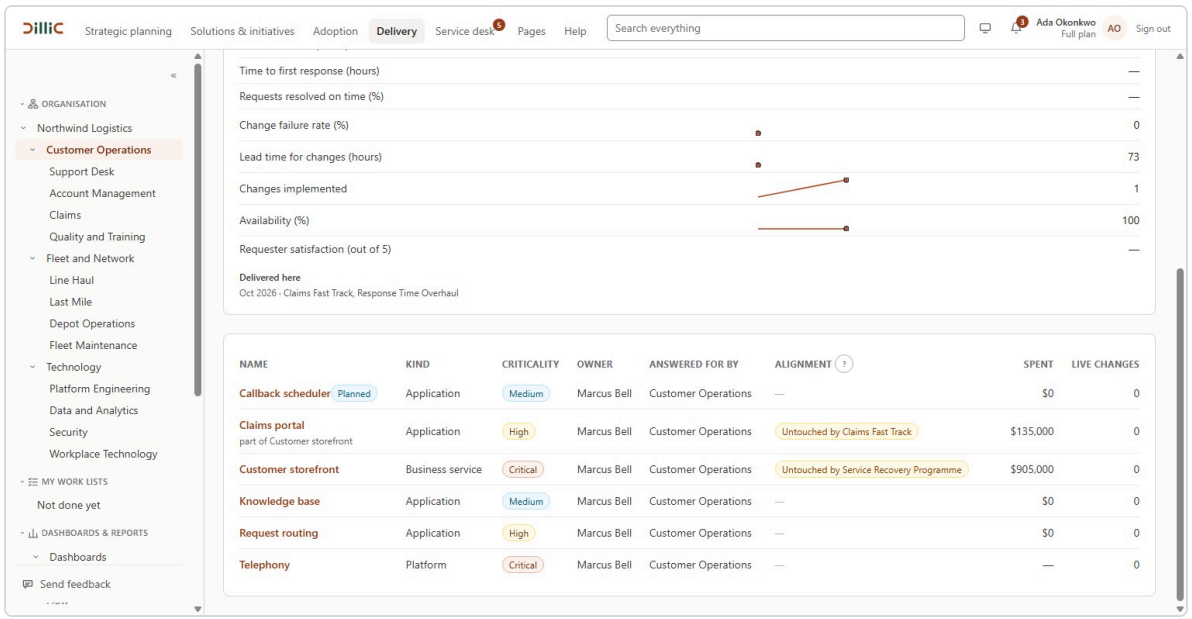


A service's page carries both halves: what the plan says about it, and what the desk and the change log say.

Services are what give a part of the organisation its value, so each service's page carries the planning half as well as the operations half:

- **Health**: the ten figures a measure/KPI can read off a service (incidents, major incidents, hours to restore, hours to first response, requests resolved on time, change failure rate, lead time for changes, changes implemented, availability, and requester satisfaction), for this service alone, a month at a time for the last twelve, each as a line with the latest beside it. A ring on a line marks a month in which an initiative delivered a change here, and **Delivered here** lists those months and what landed, so you read whether the work helped off the months after it.
- **Its plan**: what is planned for this service alone (chapter 5, Planning one service at a time): its factors with where each KPI stands, the assessments taken of it, its initiatives with their budget and spend, its risks, and its PI objectives. You see the sections you could open on their own screens, and no card where you could open none.
- The **initiatives** that establish, enhance, or retire it (chapter 8), with what each does to it.
- **Evaluated by**: the factors that evaluate it (chapter 5), each with its mark in the latest assessment that scored it.
- **Spend**: what the organisation has put into it, read from the Spend tab of each initiative that establishes, enhances or retires it (chapter 8): its budget, what is committed, and what is spent, with a total where they share a currency. Labour derived from logged hours is not counted, and an initiative you may not open is left out and said so. The list's **Spent** column gives each service's committed and spent money the same way.

- The changes against it, each saying which initiative it serves - read through the work it releases - or marked **Unplanned** when it serves none. The card's heading counts the changes of the last ninety days that serve no initiative: work nobody planned: a little is normal, but a lot on one service is worth a leader's eye.

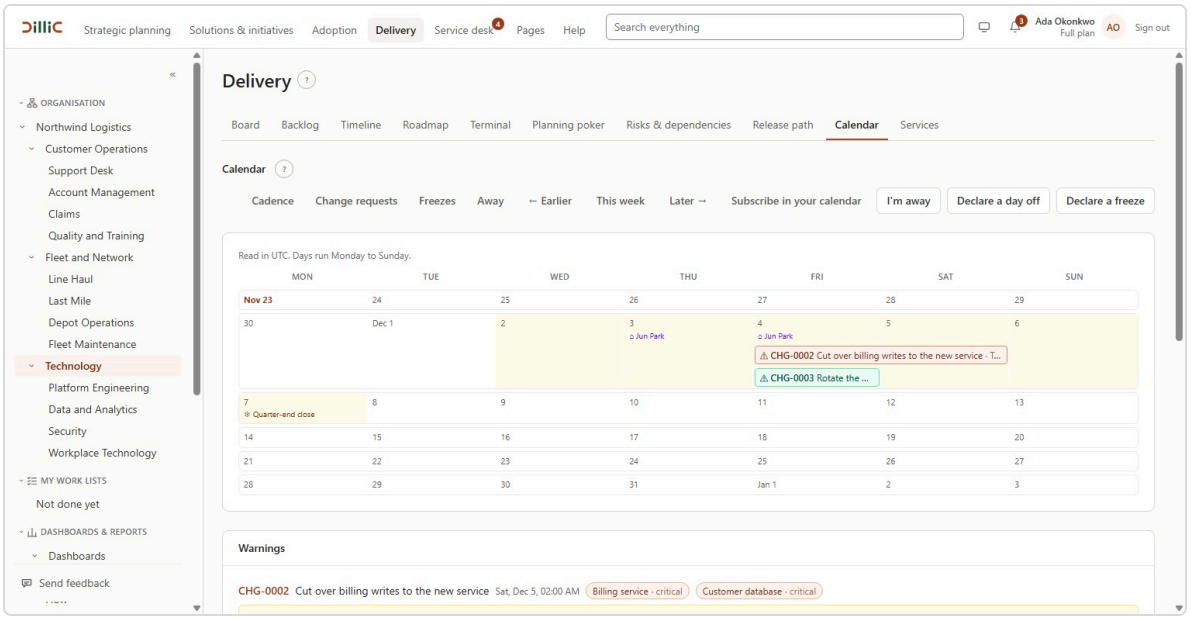


The list: the health of everything listed above it, and an Alignment column saying which services need realigning.

The list says where the plan and the delivery disagree. Its **Alignment** column carries two pills: *n unplanned*, and *Untouched by* an initiative that is under way and names this service while none of its changes has touched it. Both are read live, never stored, so aiming a release at the right change clears the pill. **Needs realigning** narrows the list to those services, and says when there are none. Above the list, **Health** reads the same ten figures over everything listed: what this part of the organisation gives the rest of it, as the health of what it answers for.

Calendar draws six weeks of planned windows as bars against the days, read from this entity and everything beneath it, with the freezes shaded behind them and the people away written into the days. **Earlier**, **This week**, and **Later** move it. A bar carries a warning when the change **collides** - another live change in an overlapping window touches one of the same services, or a service one dependency away in either direction - or falls **inside a freeze** that does not exempt its type. The same warning sits above the approval on the change's own page, so whoever authorises it reads it first. A warning is never a refusal: the change is saved and can still be approved.

Subscribe in your calendar shows your own calendar address. Add it to Outlook, Google Calendar or Apple Calendar as a subscription, and every change window and freeze you may read appears there, updated as your calendar program refreshes it. The address reads without signing in, so keep it to yourself; anybody holding it reads what you may read.



Six weeks of change windows, the freeze shaded behind them, and a warned change.

The calendar has four layers - the **cadence**, the changes, the freezes, and **who is away** - and the buttons at its top switch each off, to read the others on their own; the browser remembers which. The cadence is the sprints of the boards here and beneath, and the planning intervals here and above, drawn as thin dashed bars over the changes, so "mid-sprint" and "the planning week" are read off the same page. **I'm away** records your own days out of the office: a first day, a last day (left empty, the one day), and a note - leave, a conference, a holiday you keep. No right is needed; the days are yours. They are written into the days on the calendar of every part of the organisation you have a place on, above it or beneath it, and listed under **Who is away** with **Withdraw** on your own. It is for planning, not enforcement: a change planned for the week its implementer is on leave is still planned, but now everybody can see it.

A **day off** is when nobody is here: a public holiday, a shutdown between the years. **Declare a day off** (the same right as a freeze) takes a name, the first and last day, a note, and whether it is this part of the organisation or **The whole organisation**, declared from the top of it, exactly as a freeze is scoped. The days are shaded, listed under **Who is away** with **Remove**, and a change planned on one is warned **on a day off** on the calendar and above the approval, whatever its type - nobody being here is a fact, not a rule with exemptions. It is a warning, never a refusal.

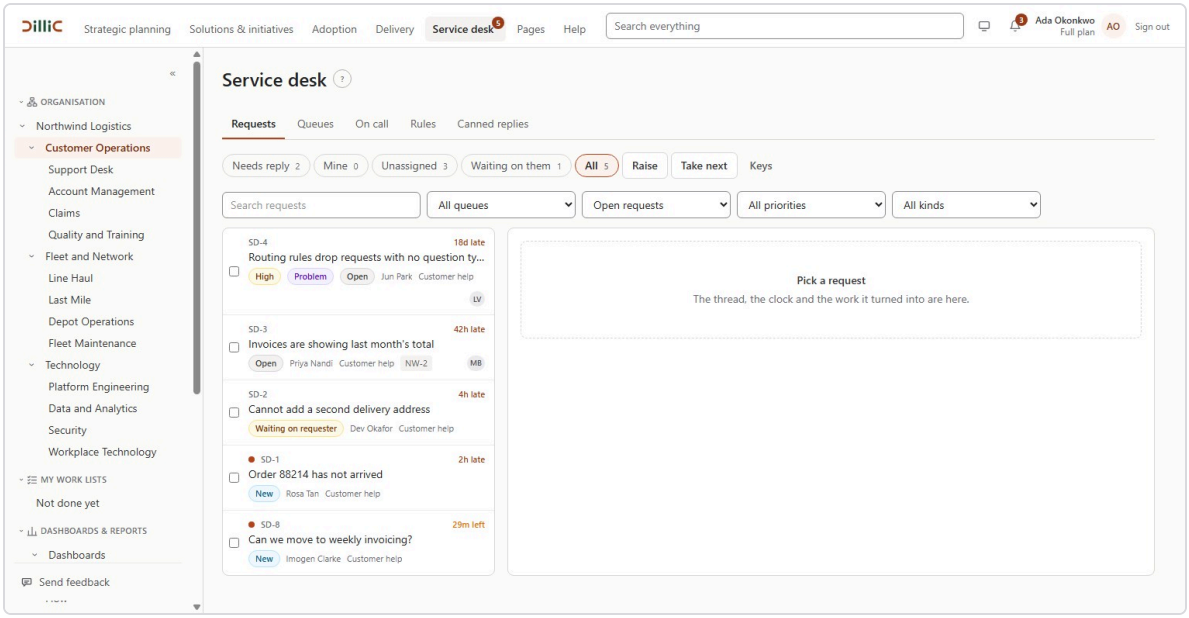
A **freeze window** is a period in which changes are warned off: the year-end close, the sale, the audit. **Declare a freeze** (right: *Declare change freezes and days off*) takes a name, a reason, the window and its time zone, whether **Emergency changes may go ahead**, and whether it is this part of the organisation or **The whole organisation**. A freeze on the whole organisation is declared from the top of it. Freezes in the six weeks are listed beneath the calendar, and **Lift it** removes one.

Rights: *See change requests* reads services and the calendar; *Keep the services a change touches* names, edits, and retires them; *Declare change freezes and days off* is the authority's. The seeded Contributor keeps services; only the Leader declares freezes.

Deployments

The organisation's build and release pipeline reports deployments to DilliC (chapter 21); nobody records them by hand. **Deployments** on the release path shows the history, filtered by environment, with each run's versions, authorising change requests, and items. DilliC flags an arrival in an environment whose authorisation is recorded when no authorised change request covers it, and on a gated environment it holds a pipeline that asks first until one does.

Service desk and portal

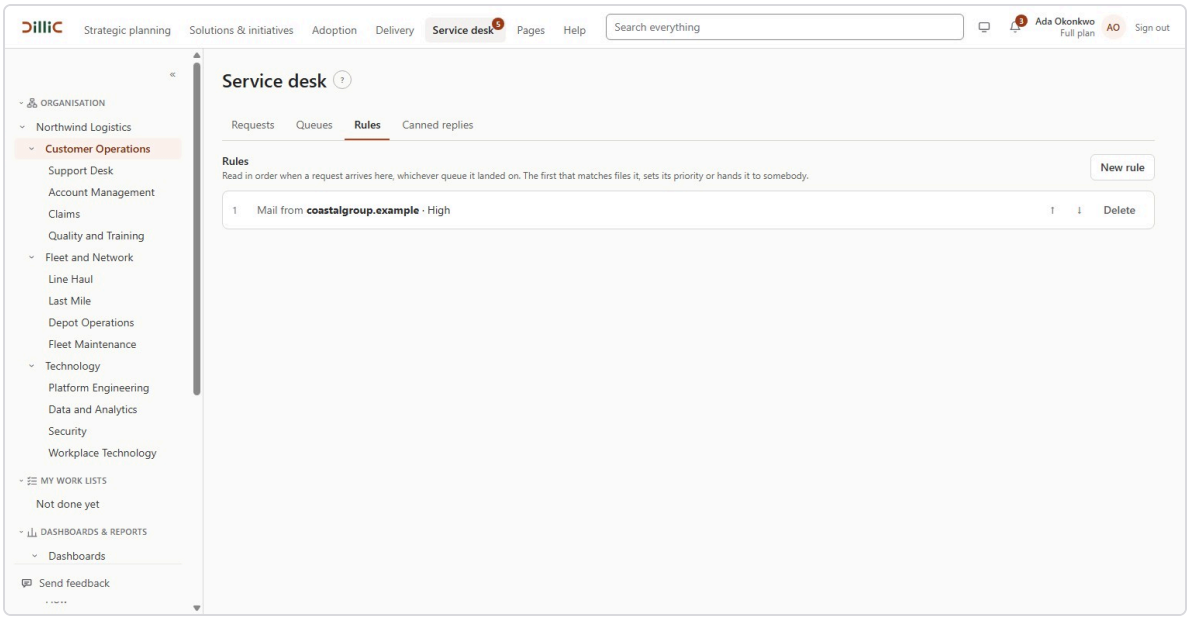


The service desk's requests: the views a shift is run from, ordered by priority and urgency.

Service desk, a tab of its own in the header beside Delivery, is where requests from outside the team arrive and where the team answers them against the times it has promised. Requesters need no account: they write through a public portal or by email. The tab carries a count of what is open, the way the bell counts what is unread, so a request arriving is seen from any screen; the header's search finds requests too, by reference, subject, or who asked, on any desk you answer. Its address is `/desk`, and `/desk?request=` opens one request, which is where a notification or a search hit lands. The desk has three views: **Requests**, **Desks beneath** (on an entity whose parts have desks), and **Queues**, with **Rules** and **Canned replies** beside them for whoever answers, and **On call** for everybody who can see the desk.

A **queue** is a promise and an address. **New queue** asks for its name, what it is for (shown on the portal), how quickly the team promises to **reply** and to **resolve**, in minutes, and optionally an **Email address** of its own. Each queue has a public portal address; click it to copy it and share it with the people who ask. The card also shows where to write by email: an address DilliC makes for the queue, which needs nothing set up, or the queue's own address with the one to forward it to beneath (Appendix D). A queue can be edited, or deleted while nothing has been raised on it; once it has history it stays, and making it private stops more arriving. The promises are stamped onto a request when it is raised, so changing a queue's promises later does not rewrite what an open request was told.

How arrivals are handled. A queue may also say: its **Categories**, a few words a requester chooses from on the portal ("Billing", "Access", "Something broke"), kept on the request; its **Assignment**, manual as every queue was, *round robin* to the next of the people who may answer, or *least loaded* to whoever holds the fewest live requests; its **Working hours**, a time zone, the days, and the hours, after which the two clocks count only working minutes - a request raised at half past five on Friday against a four-hour promise is due at one on Monday, and nothing runs out over the weekend; and its **Promises by priority**, a reply and a resolve time for each of urgent, high, normal, and low that override the queue's two numbers where one is given. On such a queue, changing a request's priority re-stamps its promised times from when it was raised, and the thread says so. The queue card says whether the desk is open now.



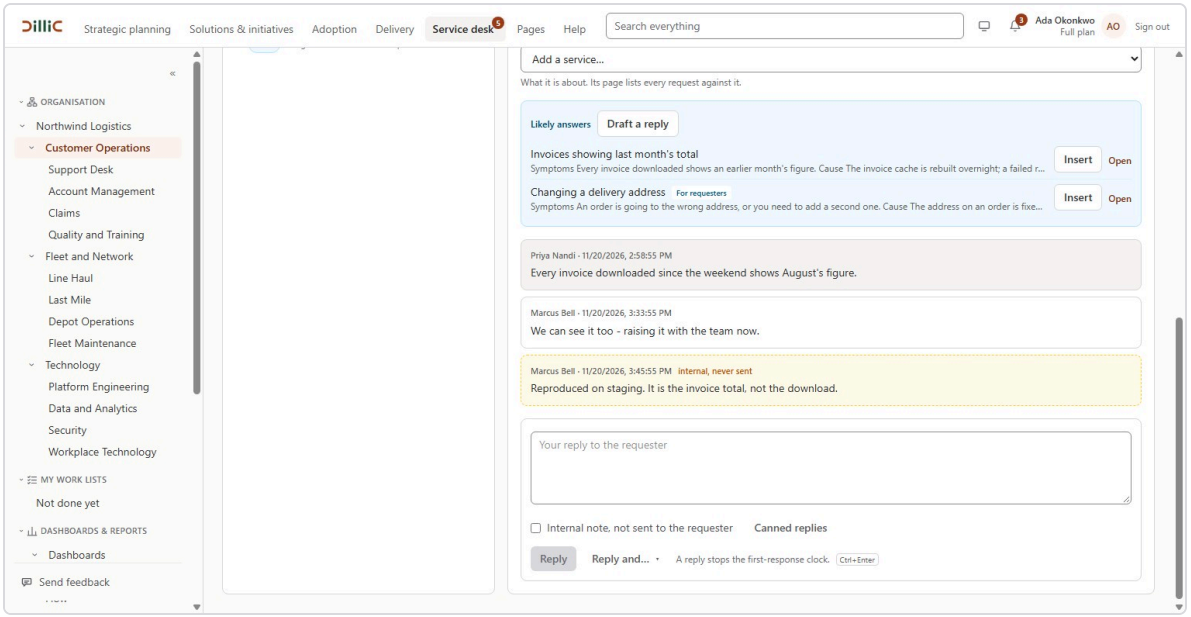
The Rules tab: the rules read on arrival, in order; first match wins.

Rules, a tab of the desk, are read in order when a request arrives on this part of the organisation, whichever queue it landed on: *mail from this domain*, *this word in the subject or body*, or *this category chosen on the portal*. The first that matches may file the request on another of the entity's queues, set its priority, and put somebody's name on it; a rule has to do at least one of the three. All of it happens before anybody is told, so the people notified are the ones the request ended up with, and the promised times are stamped from where it ended up. The thread notes what each rule did, as a note from the desk. ↑ and ↓ reorder rules, since first match wins.

Take next, beside the views, puts your name on the most urgent request nobody holds - highest priority first, then closest to falling due - and opens it; `t` does the same. It says so when nothing is waiting.

Requests open on **Needs reply**: everything where the requester had the last word, which is what a shift starts from. Beside it sit **Mine**, **Unassigned**, **Waiting on them**, and **All**, each with its count. A dot on a row says it needs a reply, and initials say whose it is. Within a view the list is ordered by priority, then by urgency, each row with a clock: time left, met, late, or met late, turning amber under an hour. Filter by queue, by status, and by priority, or search on the reference, subject, requester, or queue; naming a status outright switches the view to All. Open a request to:

- set its **Status**: New, Open, Waiting on requester, Resolved, or Closed;
- set its **Priority**: Low, Normal, High, or Urgent. Normal is unmarked; the others show on the row;
- choose who it is **Answered by**, or **Take it** yourself; that person is then the one told about it;
- change the **Queue this belongs to**, to any queue you answer anywhere in the organisation. The promised times stay as they were made, the new queue's desk is told, and a note on the thread says who moved it, from where, and when;
- **Reply**, or **Reply and…** resolve or set waiting in the same breath; or tick **Internal note, not sent to the requester** to talk among the team. **Canned replies**, or `/` in an empty box, inserts one the desk has written with the requester's name, the reference, the queue, and your name filled in;
- **Link the work that resolves it**, or **Create a work item from this**: an ordinary item on the desk's entity, titled with the subject and carrying who asked and what they first said, tied to the request. The thread notes it.



A request open on the pages that sound like it, with Insert and Open beside each.

Likely answers. Where the organisation has already written something that sounds like the request - a page on the desk's entity or the ones above it, matched on the subject and what the requester last said - the request shows up to five, best first, each marked *For requesters* where the page is one they may read, *Overdue for review* where its date has passed, and *Already sent* where it has been handed to this requester before. **Insert** (or `i` for the first) puts the page's Fix into the reply - with its title and portal link where the requester may read it - and remembers that it was handed over; **Open** reads the page. Once a request is resolved or closed, **Write a page from this** turns the question and the last reply into a page filed under the queue's answers, for next time. Pages says how a page becomes an answer.

What the assistant suggests. On the full plan, the desk has an assistant, and everything it says is a suggestion: nothing moves and nothing is sent until an agent says so. Every few minutes it reads the requests nobody has looked at - a few per tick, and only while the organisation has credits to spend - and a request then opens with a two-sentence *summary* under its subject and a row of *Suggested* chips: a priority,

a better queue where one fits, and the language the requester wrote in. **Apply** takes one, **Apply all** the lot, and **Dismiss** puts them away for that request; **Ask the assistant** asks on demand where it has not looked. In *Likely answers*, **Draft a reply** asks it which of those pages answers the request and writes the reply from the best one into the reply box, with a line beside each page saying why it ranked it; where none fits, **What next?** says so and offers the next step - **Ask** with a clarifying question drafted, **Move**, **Create a work item from this**, or **Escalate**. DilliC shows the assistant only the pages the desk found; a page it was not shown cannot be cited, and a draft with no page behind it is not offered. Each question costs AI credits from the organisation's allowance, like a report's analysis; when the model cannot be reached, the built-in matcher answers, costs nothing, and the card is marked *Offline*.

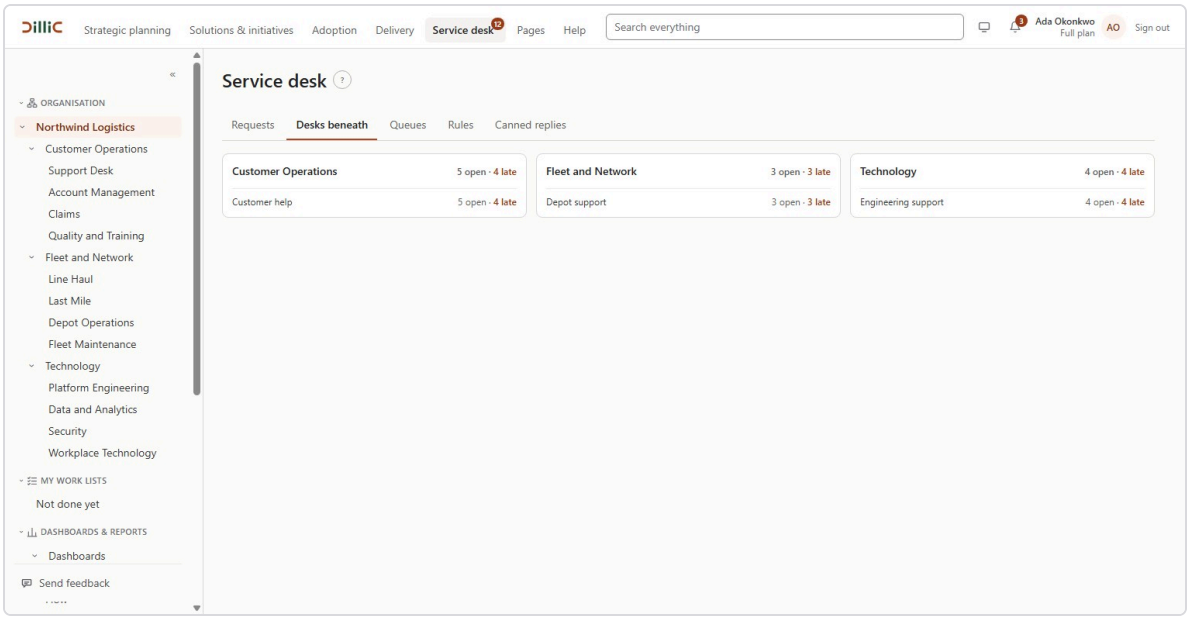
The first public reply stops the reply clock. Waiting on the requester pauses both clocks; resolving stops the second. A requester replying to a waiting or resolved request reopens it; a closed one takes no more replies. When a request leaves the list - resolved, moved, handed on - the desk moves on to the one that stood next to it, so a morning's requests are worked without reaching for the mouse.

Keys. With nothing focused, `j` and `k` move along the list, `Enter` or `r` starts a reply and `n` a note, `Ctrl+Enter` sends, `e` resolves, `w` sets waiting, `o` reopens, `a` takes it, `t` takes the next one waiting, `u` unassigns, `1` to `4` set the priority from urgent to low, `m` opens the queue control, `x` selects the row, and `Esc` clears the selection or leaves the reply box. `?` shows the map; so does **Keys** beside the views.

Several at once. Tick rows (shift-click runs a range, `x` ticks the one open) and a bar appears: **Set status**, **Set priority**, **Move to queue**, and **Hand to** apply to all of them as they are chosen, with the same notes, letters, and notifications a change to one carries. DilliC skips and counts a request you may not answer, rather than stopping the rest.

Canned replies are kept on their own tab of the desk by whoever configures the desk: a title to choose it by and the words, which may carry `{name}`, `{reference}`, `{queue}`, and `{agent}`. Written on a part of the organisation, a reply is offered on every desk beneath it, so a company's set reaches every team and a team's own stay its own.

Being told. Everybody who may answer on the queue's entity is notified when a request arrives and when its requester writes back; once a request has somebody's name on it, that person is told alone. Every few minutes DilliC reads the clocks and tells the same people once when a promise is close - half an hour ahead, or half the promise if that is less - and once when it has passed. Pressing the notification opens the desk on that request. The kind to tick in your settings is *When a request on a desk I answer arrives, is written back on or is running out of time*.

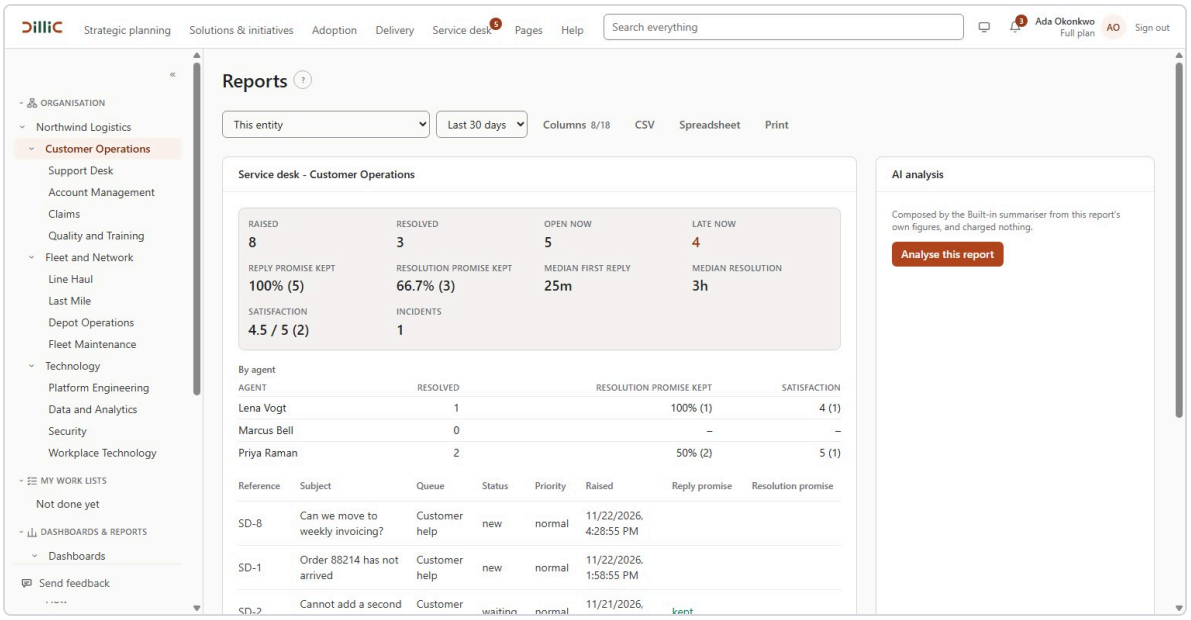


A parent's desk: the desks beneath it, with what is open and late on each.

Desks beneath. On an entity whose parts have desks, this tab lists each part that has a queue somewhere under it, with what is open and late across the whole of it now: its own queues one by one where it has them, or the part alone with its total where its desks are further down. The tab leaves out a part with no desk anywhere beneath it. Pressing a part goes to its desk. The desk opens on this tab when the entity has no open requests of its own, and the count on the Service desk tab in the header includes everything beneath. An entity with no delivery board of its own still has its desk, since the desk is not a view of a board.

The portal needs no account. It carries the organisation's logo where its desk's entity has one, so requesters know whose desk they are writing to. They give their email, optionally their name, *What do you need?* and more detail, choose a category where the queue offers any, may **Attach files** - up to six, 10 MB each - and **Send it**. The portal shows them the team's promises and emails them a private link to follow the request, where it reads "With the team", "Being worked on", "Waiting on you", "Answered", or "Closed", where what they attached is named, where the pages the team pointed them to are listed, and where they can add to it or attach more. They never see internal notes. A closed request takes no more replies or files; they raise a new one. As they type what they need, the portal offers up to three pages marked for requesters under *This may already answer it*; each opens on the portal, with *Did this answer it?* at the foot - a yes is a request never raised, and **No, raise a request** returns to the form with what they typed kept.

By email is the other door. Each public reply from the team, and the word that a request is resolved, goes to the requester as a letter, headed with the organisation's logo where it has one. The letter carries the request's own reply address, so answering is pressing Reply. A letter sent to a queue's address raises a request on it, with its attachments, and is acknowledged with the portal link; a reply to a closed request opens a new one. Replies that arrived by mail are marked *by email* on the thread. Appendix D says which address to hand out, and how to use one of your own.

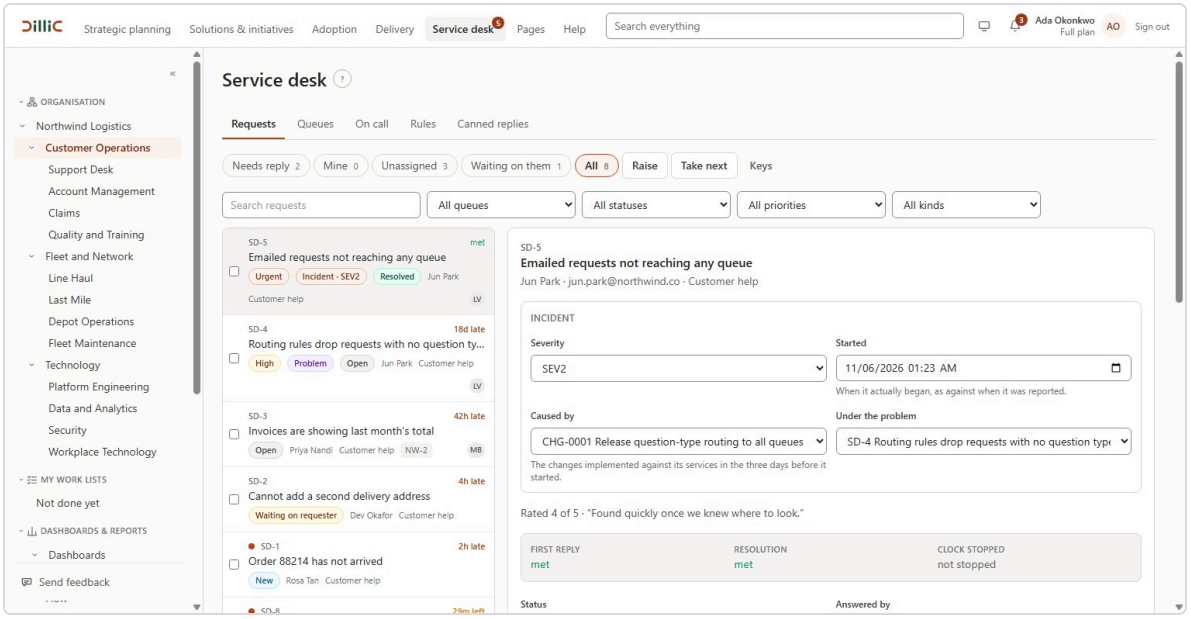


The service desk report: how the promises were kept, overall and by queue.

The report. Reports holds a **Service desk** report: over the last 7, 30, or 90 days, what was raised, resolved, open and late now, how often the reply and resolution promises were kept, and the median first reply and resolution, overall and by queue, with every request the period touches as a row. See the Reports chapter.

The module. The service desk is a module inside Change management (Admin → Modules). When it is switched off, the tab, the report, the right to answer requests, the request notifications, the portal, and the mailbox all go together; nothing is deleted.

Incidents, problems and ratings



An incident: its severity, when it started, the change that caused it, and the problem it sits under.

A thing on the desk is a **request**, an **incident**, or a **problem**, and the list filters by kind. An incident is something that has broken: it carries a **Severity** (SEV1 to SEV4), when it **Started**, the services that went down, the change it was **Caused by** - offered from the changes implemented against those services in the three days before - and the problem it sits **Under**. A problem is the reason behind one or more incidents: a **Root cause**, a **Known error** saying what to do until the fix ships, the incidents under it, and the change it is **Fixed by**.

1. **Raise** on the desk raises a request, an incident, or a problem from inside the organisation; you stand as the requester. Or raise an incident from the change that caused it (Change requests, above).
2. A SEV1 or SEV2 incident on a service tells the service's owner, through their notification switches, whether or not they answer the desk.
3. A service's page lists every incident and request against it; a change's page lists what it caused.
4. A root cause analysis (chapter 7) cites incidents and problems as evidence for an assertion or a cause, and is told which factors are worth assessing next time.

Ratings. When a request is resolved, the letter that says so carries one link that rates how it went, from one to five with a line, once; the request's own portal page offers the same stars while it may be rated. The desk sees the rating on the request, and the Service desk report carries **Satisfaction** overall, by queue, and by agent.

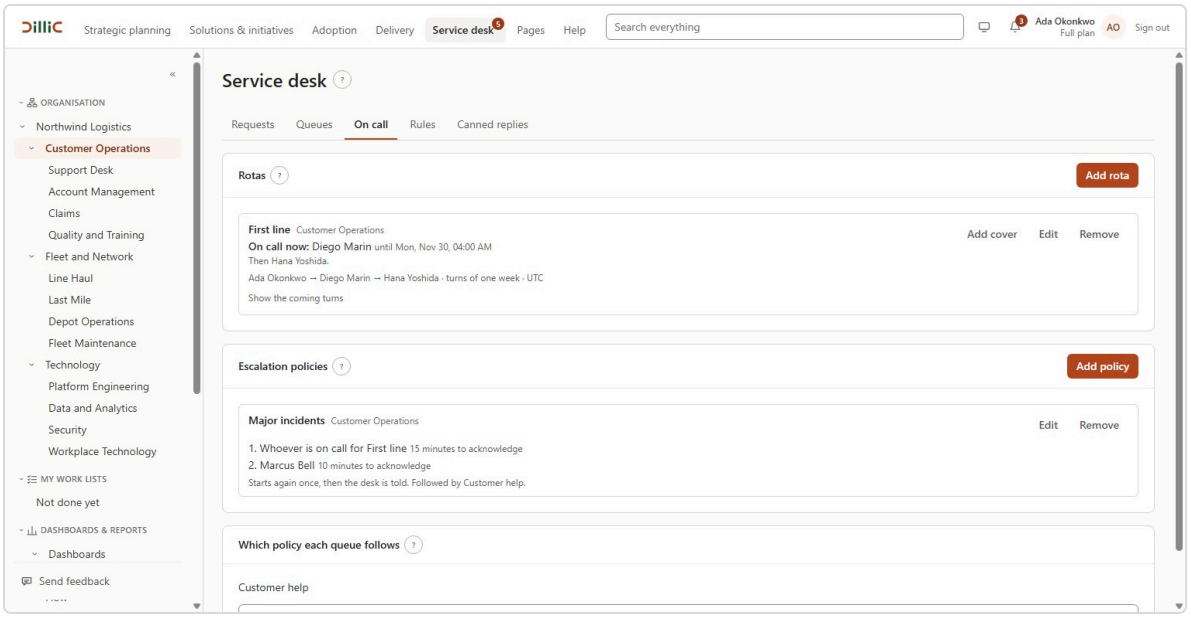
An incident from a monitor, or from another desk

A monitor that already knows a service is down can raise the incident itself, and so can a service desk the organisation keeps elsewhere. An account administrator opens Administration, then **Alerts in** under Connections, and adds a source: its name, the queue its incidents land on, what it sends, and the severity an alert gets when it names none. The screen shows the source's address and its secret once; paste both into the sender's webhook settings.

1. **What it sends.** **Grafana** reads Grafana's alerting webhook as Grafana sends it. **PagerDuty** reads PagerDuty's webhooks, and checks the signature PagerDuty writes with the secret. **DilliC's own shape** is for anything else, another desk's automation included: JSON with a `fingerprint`, a `title`, a `description`, a `severity` from sev1 to sev4, a `status` of firing or resolved, a `service` and a `url`.
2. **The secret.** The sender proves it holds the secret with a bearer token, or with the `X-DilliC-Alert-Secret` header. A wrong one is refused before anything is read. **New secret** replaces it at once.
3. **What follows.** An alert firing for the first time raises an incident on the service it names, matched by name, with its severity and its link. The same alert firing again is a line on that incident, not a second one. A resolved alert resolves it. A SEV1 or SEV2 wakes the service's owner, as one raised by hand does.

DilliC detects nothing itself. An alert naming a service the desk does not keep raises the incident without one, and says what the alert called it.

On call: rotas, escalation and pages



On call: whose turn it is, the policy a major incident pages, and the queues that follow it.

On call on the desk says who is woken for a major incident. Anybody who can see the desk reads it; whoever may configure the desk keeps it.

1. **A rota** is an order of people who take turns. **Add rota** asks for a name, when the first turn begins, the time zone the handover is agreed in, how long a turn lasts, and the people in order. Nothing is drawn by hand: who is on call at any moment follows from those. **Show the coming turns** lists the next six weeks.
2. **A cover** puts somebody else on call for a stretch. Whoever keeps the rota adds one with **Add cover**; somebody on the rota takes one with **Take a cover**. Somebody who has marked themself away on the change calendar is passed over for the next person in the order, on the days they are away.
3. **An escalation policy** is who is told, in order. Each step names a rota, which means whoever is on call for it at that moment, or one person, and gives them so many minutes to acknowledge. When the last step goes unanswered the policy starts again as often as it says, and then everybody who answers the desk is told.
4. **What follows a policy.** Each queue chooses one under **Which policy each queue follows**. A service may have its own, set on the service's page, and it is read before the queue's.

A SEV1 or SEV2 incident pages its policy the moment it is raised, whether a person raised it or an alert did, and so does an incident whose severity is raised to one of those. On any request, an agent may use **Page on call**. The request shows who was paged and when the next person will be. **Acknowledge** ends the page; the request takes your name if it had nobody's. Resolving the request ends the page too.

A page goes by email, by Slack or Microsoft Teams where you have connected one, and to the phone app, at once and whatever your notification settings say. DilliC does not telephone and does not send text messages.

A request from Slack or Microsoft Teams

People may raise a request without leaving their chat tool. An account administrator opens Administration, then **Chat in** under Connections, and adds a door: its name, the chat tool, the queue its requests land on, and the secret the chat tool signs with. The screen then shows the door's address.

1. **Slack.** Add a slash command to a Slack app and give it the door's address as its request URL. The secret is the app's signing secret.
2. **Microsoft Teams.** Add an outgoing webhook to the team with the door's address as its callback URL. The secret is the security token Teams shows once.
3. **What people type.** Words raise a request: the first line is the subject. `status` lists what you have open. A reference such as `SD-12` says how that request stands, and a reference followed by a message adds the message to it. `help` explains.
4. **What comes back.** The person is answered at once, and only they see it: the reference, the queue, the time a first reply is promised in, a link to follow the request, and up to three help pages that read like the question.
5. **The desk's answers.** Give the door the channel's incoming webhook and each public reply, and the resolution, is posted to the channel. Internal notes are never posted. Without a webhook, the requester follows the link.

A request raised this way passes through the queue's rules as any other does, and the desk marks it and its messages *by chat*. The chat tool gives DilliC no email address, so no letters are sent for these requests.

Planning poker

Whoever leads uses **Open a session** and picks the next unsized item. Everyone in the room picks a card: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, or ?. Nobody, the lead included, sees a vote until everyone has voted and the cards turn over by themselves, or the lead chooses **Turn the cards over**. Talk about the spread; then the lead writes down the agreed estimate. **Step out** leaves the room; people without a vote watch.

Pages

Pages, a tab of its own in the header, is the team's wiki, kept with the work: a tree on the left, including the pages of teams beneath, and the page on the right. **Write a page** or **Write a page inside** another; a page can say which work items it is about. **What it said before** lists earlier versions to read or put back. **Retire it** hides a page without losing it.

Every initiative and every epic has a page of its own, blank until somebody writes it. An epic's page is filed inside its initiative's, and an initiative beneath another - a SAFe capability under its epic - has its page filed inside the parent initiative's, so the tree of the entity above reaches down through the hierarchy. The page says what it is the page of in your organisation's own word for it: a feature, on a train. These pages are named by what they are the page of and follow it: rename or archive the initiative, and its page follows; retire the page, and the initiative is archived. The page carries a link to the initiative or epic, and the initiative and the epic each carry a **Page** link back. The tree folds like the left pane: the « above it folds it to a rail,

resting the pointer on the rail unfolds it over the page, and every branch remembers whether it was open. Hovering a row shows a star: pin a page and it stays at the top of the tree under **Pinned**, from whichever team it belongs to.

A page as an answer. A desk answers the same questions all year, and the answers are pages: there is no separate knowledge base. Under a page's settings, tick **Also for requesters: offered on the desk's portal, to anybody with the link** and the page is served under the portal of every queue on its entity and the ones beneath, offered as a requester types, and inserted into replies with its link; a page the team keeps to itself is still offered to agents and quoted into replies in the team's words. Who may *edit* a page never changes - that stays the entity's roles. Inside a requester-facing page, links to other pages are kept only where those are for requesters too; mentions, work items and uploaded pictures appear as their words. The **Article** template - Symptoms, Cause, Fix, Applies to - is the shape for one, and the Fix is what the desk quotes. A page that has been handed to requesters says so at the top: *Handed to 40 requesters; 31 said it helped*. The desk's own **Write a page from this** files a new one under *Queue: answers* on the queue's entity, for the team until somebody marks it for requesters, with a review date six months out.

Pages go out and come in. **Export** on a page saves it as Markdown - one file, or an archive with the pictures beside the pages when you take the pages inside it too - or makes a PDF through your browser's print dialog (choose **Save as PDF**), with a table of contents for a tree. **Import** on the Pages heading takes Markdown files, a zip of them whose folders become pages inside pages, or a Confluence HTML space export, at the top or inside the open page. Links between the imported pages point at the pages they became, pictures are uploaded onto their pages, and Confluence's code panels, info boxes, task lists, and status lozenges arrive as the plain shapes they meant. On the phone a page is written in the same editor as on the web.

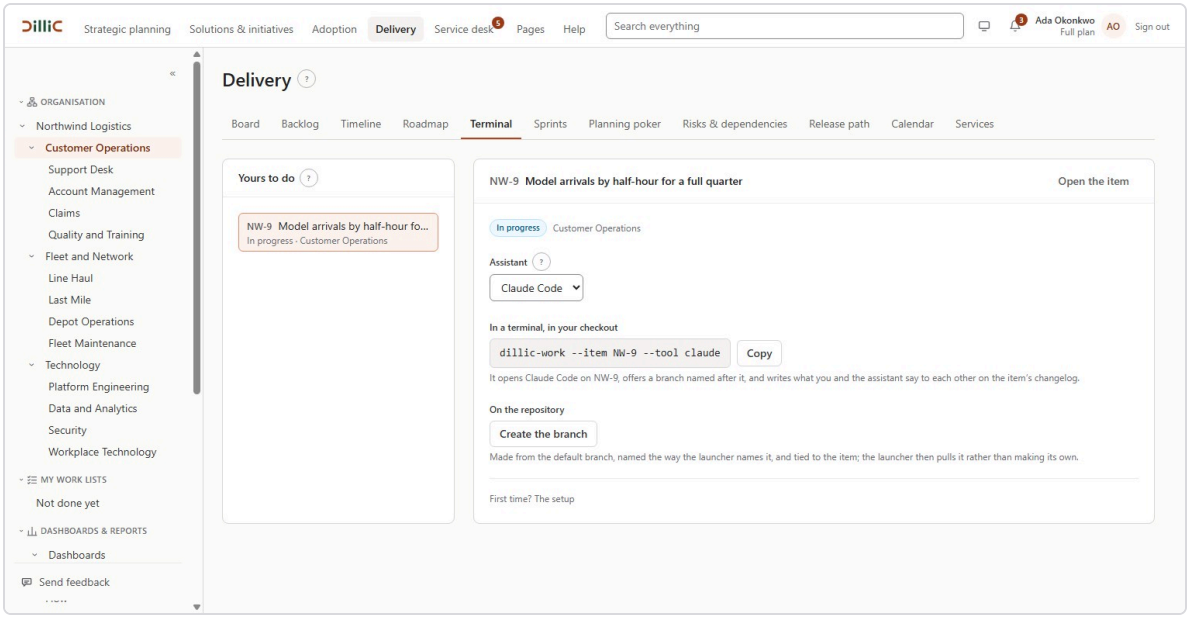
A page links to other pages and knows what links to it: type `[[` in the editor to link a page, and every page lists **Links to** and **What links here**. Type `#` to link a work item, or type only its key (`NW-12`): it is drawn live, with its title and where it has got to. A saved list on a line of its own shows its rows. Paste or drop a picture into the page, or use **Image**; a table comes from **Table**. Type `@` to name somebody, and they are told. Every page carries a **Discussion**, and **Watch** follows it, so you hear when it changes or somebody comments; writing or editing a page follows it for you.

A page names an **Owner** and a **Review by** date. Once the date has passed, the page is marked *Overdue for review* wherever it is listed, and the owner is told once; saving it with a new date clears it. **Labels** are a few filing words, shown as chips above the tree to narrow it, and **Search pages** finds a page by title, label, or words in its body. **From a template** starts a page from a shape - a runbook, a decision, meeting notes, an incident, a how-to - or from any page the team has ticked *Offer this page as a template*. A page with three or more headings gets a **Contents** list. In **What it said before**, **What changed** shows a version against the one that replaced it, word by word. If somebody else saves a page while you are editing it, DilliC refuses your save, and you choose: take their version, or save yours over it.

Working with an AI assistant

If you write code with an assistant such as Claude Code, Cursor, or Codex, the **Terminal** tab lists everything assigned to you and not done, across every team you can see. Pick one and choose your assistant, and DilliC

gives you one command to paste into a terminal in your checkout. It opens the assistant on a branch named after the item, and every exchange is written on the item's changelog, marked with the tool's name.

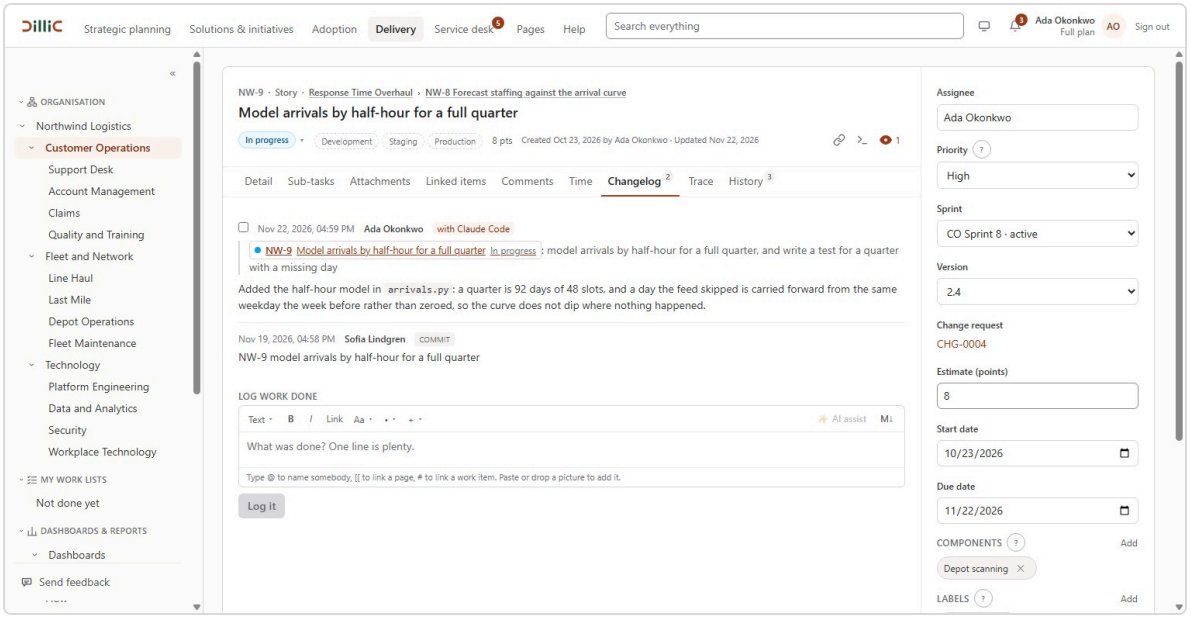


The Terminal tab: pick an item and an assistant, copy the command.

First time, under **First time? The setup**:

1. Create an access token (chapter 2) that can view, edit, and move work.
2. Put the `dillic-work` launcher on your path.
3. Run `dillic-work login` and give it DilliC's address and the token.

An item nobody has started is offered its next step first, because nothing is logged on work that has not begun. **Create the branch** makes the branch on the repository instead, where one is connected.



The result on the item: the question, the answer, and "with Claude Code".

Delivery above the teams

An entity with teams beneath it has no board of its own. Its Delivery page reads theirs, one level up at a time: at a division, the board and backlog show the teams' epics, and the timeline and roadmap show initiatives; one level higher, the board shows initiatives and the timeline shows solutions. Columns are grouped by what a status means, since teams may name theirs differently, and nothing can be dragged. The release path shows every change request raised beneath.

CHAPTER 10

Portfolio and scaled planning

Where an organisation runs SAFe (chapter 19), two more areas appear: **Portfolio**, a tab of its own in the header where initiatives are funded, and **Planning intervals**, a tab of Delivery where teams on a train plan together. The portfolio is also its own module, so an organisation can fund epics here without running its boards here.

The portfolio kanban

The portfolio shows initiatives as epics moving through Funnel, Reviewing, Analyzing, Portfolio backlog, Implementing, and Done (or Cancelled), ordered within each column by their WSJF score. Each card shows its hypothesis, the value stream funding it and what that stream has left, its budget, and its score. Those who manage the portfolio can:

- write its **Business case**: the epic hypothesis, business outcomes, leading indicators, MVP, non-functional requirements, and the value stream it is funded from;
- score it: business value, time criticality, and risk reduction, over job size;
- move it a column on or back. Entering the portfolio backlog is gated by **Lean Portfolio Management**: two approvers, not the proposer.

"Not on the portfolio yet" lists initiatives nobody has proposed.

Value streams

A value stream funds epics. Whoever manages the portfolio creates one with **New value stream**: its name, what it funds, its budget, and its guardrail. Each shows its budget, what is committed, proposed, and spent, what remains, and a warning when over. Its **guardrail** is soft (warn when committed exceeds the budget) or hard (refuse a funding move the stream cannot afford). A stream and the initiatives it funds must share a currency.

Planning intervals

1. **New planning interval**: a name ("PI 26.1"), start, iteration length and count, a goal, and whether the last iteration is for innovation and planning. **Plan it**.
2. **Generate iterations** puts every team on the train onto the same dates. Teams whose sprints already overlap are skipped and named.
3. Each team writes **objectives**: the objective, its business value (1–10), and whether it is committed or stretch.
4. Take a **confidence vote** (1–5) and record the interval's risks.
5. **Start** locks the commitments; wording can still be corrected.

6. At the end, enter the value each objective delivered and **Close**. Closing is refused while a committed objective has no delivered value.

The interval reports **predictability** (delivered against committed value; stretch objectives count only towards delivered), points per iteration, feature cycle time, flow efficiency, and what the interval was spent on.

The program board

The program board sets teams down the side and the interval's iterations across the top. Each feature sits in the earliest iteration its work is planned in. Dependencies between items are drawn as dashed lines; one that runs backwards in time, where the needed work lands after the work needing it, is drawn in red and counted in a warning. Features not yet in an iteration are listed beneath. The board reads the plan; change it on the teams' boards.

CHAPTER 11

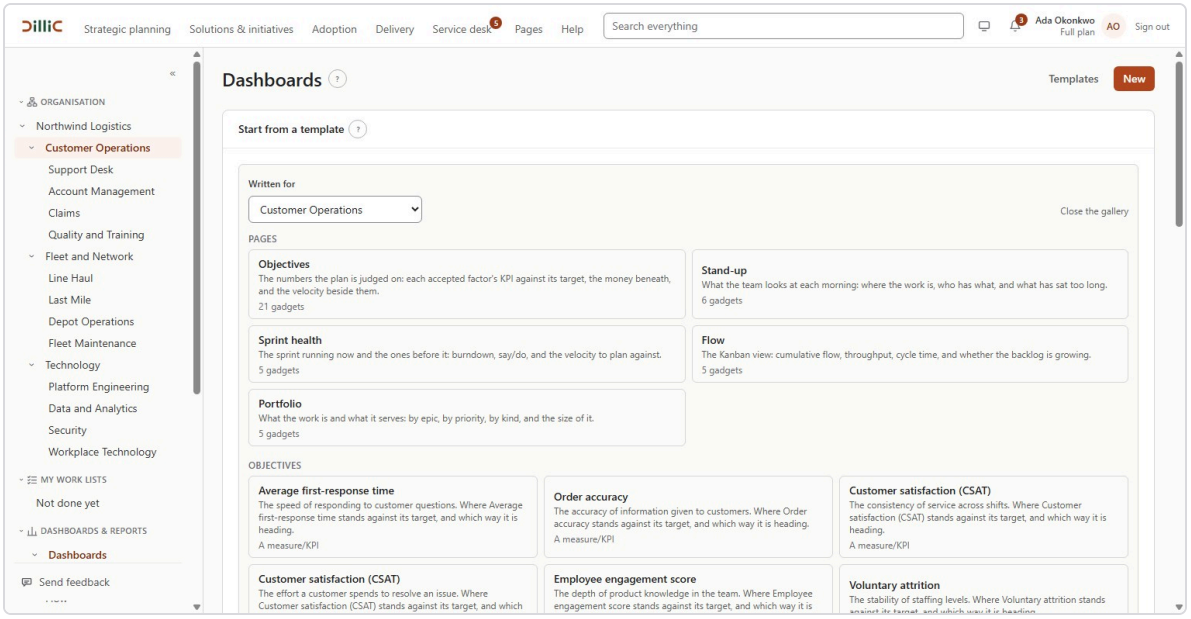
Dashboards and reports

Dashboards

A dashboard is a page of questions answered every time you open it. Each **gadget** on it is a question you can edit. The **Dashboards** group in the left pane lists yours and those shared with you; click one to open it. Beside the heading of the Dashboards screen, **Templates** opens the gallery and **New** starts a blank page.

1. Start from a template page (Objectives, Stand-up, Sprint health, Flow, Portfolio) for the board you choose, add single gadgets from the gallery, or start blank.
2. Name it (**Called**) and say what it is for.
3. Adjust each gadget: heading, what it is about, which entity, how it is drawn, and its width (a third, two thirds, or the row).
4. **Share it** if others should see it, and save. A shared page is answered for whoever is reading it, so two people see two correct sets of figures and nobody is shown work they may not see.

Only the owner edits or deletes a page; anyone else can **Copy it to change it**.

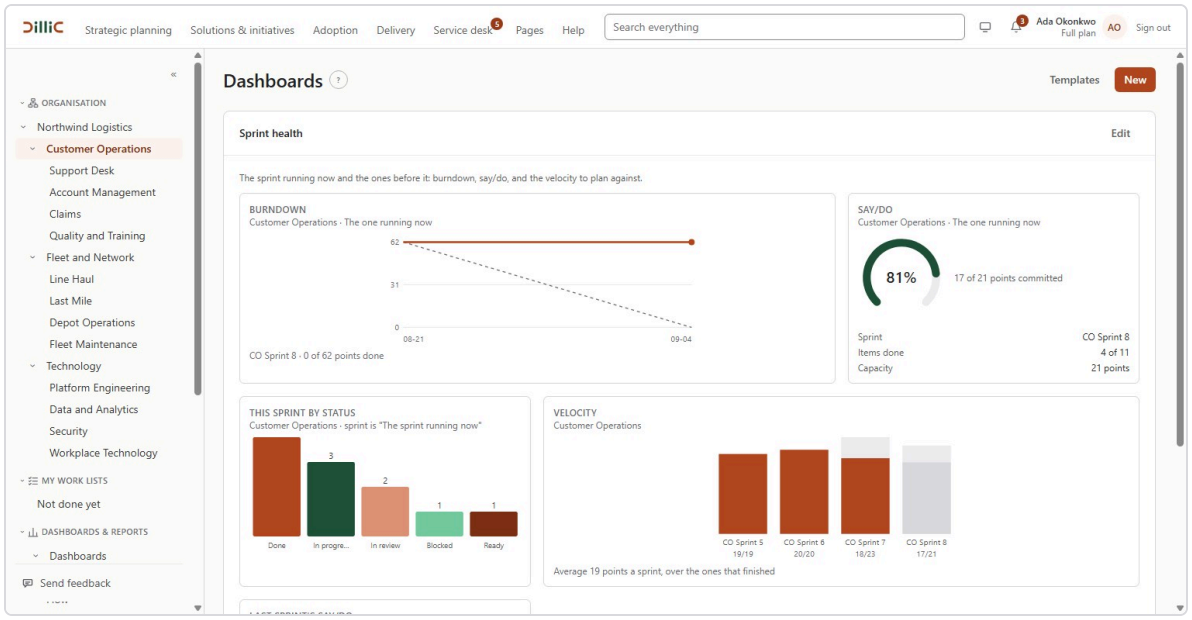


Start from a template: pages first, then single gadgets by shelf.

Gadgets

GADGET	SHOWS
A number	How many items, or points, answer a question.
A short list	The first few rows that answer it (up to 20).
A breakdown	The answer counted by one field, as bars, columns, a donut, or one stacked bar.
A sprint's say/do	What a sprint took on against what it finished, as a gauge.
A burndown	Points left per day against the ideal.
A velocity run	Points finished in each of the last sprints.
Throughput per week	Items (or points) finished each week.
A cumulative flow	Waiting, in progress, in review, and done, day by day.
Cycle time	How long work took, as a histogram with the median and the flow efficiency.
Aging work	What is started and not finished, oldest first.
Arrivals and departures	Work created against work finished, per week.
A KPI	One KPI's current value against its target, for an entity alone or with everything beneath.
Money	Budget against spend, for everything beneath, one initiative, or one value stream.

A gadget can be about work, factors and objectives, value streams, initiatives, change requests, planning-interval objectives, assessment results, or time spent. Sprint gadgets ask about the sprint running now, the next, the one after, or the one just finished; flow gadgets look back four weeks to a year.



Sprint health: burndown, say/do, and velocity.

Template pages

Objectives	Each accepted factor's KPI against its target, the money beneath, and velocity.
Stand-up	Where the work is, who has what, what has sat too long, mine, critical and open, blocked now.
Sprint health	Burndown, say/do, this sprint by status, velocity, last sprint's say/do.
Flow	Cumulative flow, throughput, cycle time, arrivals and departures, oldest in flight.
Portfolio	Epics taking the most of the team, work by priority and kind, open work, points in flight.

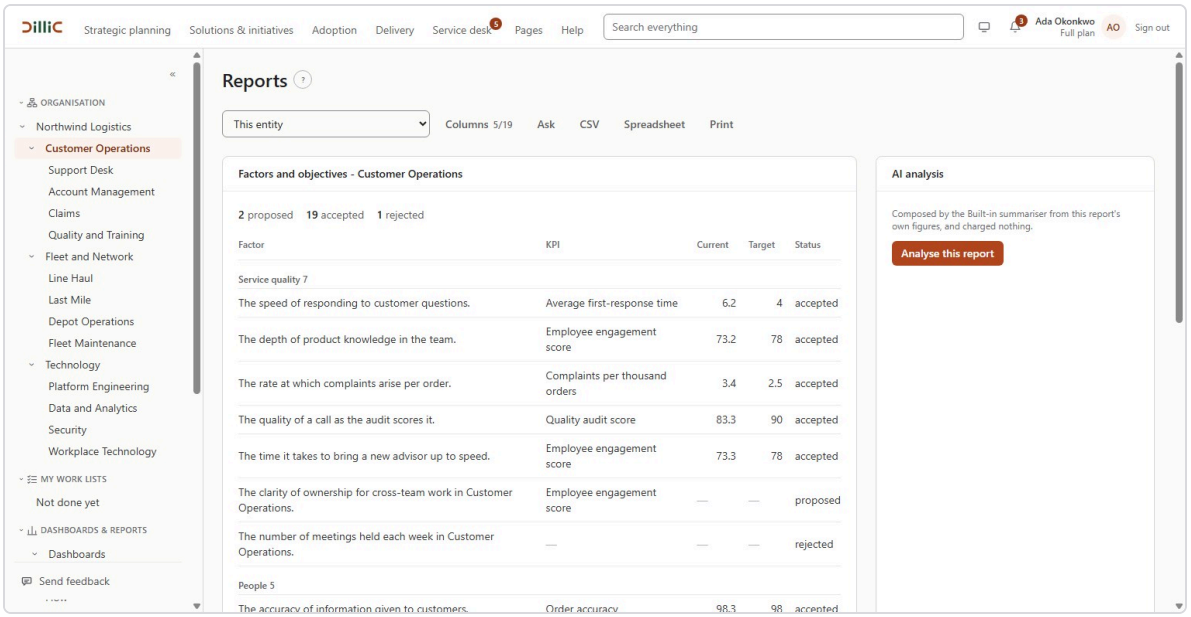
Templates are written in the chosen board's own status names, and a gadget the board cannot answer is left out.

The shipped reports

Reports in the left pane holds five reports for whichever entity is chosen, for that entity alone or with everything beneath it:

REPORT	SHOWS
Factors & objectives	Every factor with its objective, KPI, current value, target, trend, readings, votes, applicability, and status; and the KPIs beneath.
Assessment	One assessment's marks: scores, agreement, quadrant, responses, and spread, and the combined score.
Root cause analysis	One analysis's assertions and causes with their status and factor.
Service desk	Over the last 7, 30, or 90 days: requests raised, resolved, open and late now, how often the reply and resolution promises were kept, the median first reply and resolution, satisfaction and incidents, overall, by queue, and by agent; with every request the period touches as a row. Needs the right to answer requests.
Change requests	Over the last 30, 90, or 180 days: changes implemented, the success and change failure rates, the share that were emergencies, the median lead time from raising to done, the incidents they caused, and deployments to production per week - the DORA pair beside the third - by month and by service; with every change the period touches as a row. Needs the right to see change requests.

A report is headed with the organisation's logo where the entity has one, on screen and in print. **Columns** chooses what is shown (remembered in this browser, and what prints and downloads). Every report can be downloaded as **CSV** or **Spreadsheet**, or printed. **Ask** asks a one-off question on the same subject without saving it.



The factors and objectives report, with AI analysis beside it.

Writing your own report

1. **Write a report**. Name it ("Unestimated work by division") and say what it is for.

2. Choose what it is **About** and **Which part of the organisation**: everywhere you can see, or an entity and everything beneath it.
3. Add **conditions** (the same tests as lists, "match all" or "any"). A person can be **Me**: whoever is reading.
4. Optionally **Grouped by** a field, and **Drawn as** bars, a line, or a donut. The table always shows underneath.
5. Add **totals**: how many, total, average, lowest, or highest of a field.
6. Choose columns and sort, tick **Share it with everybody who could have written it** if you like, and save.

Your reports appear under **My custom reports**; shared ones under **Shared custom reports**, where others can read them or copy them to change. An answer that reaches the ceiling stops there and says its figures are a floor rather than a total.

AI analysis

On the full plan, a report has an **AI analysis** panel. **Analyse this report** reads the report's own figures and returns a summary, insights, and recommended next steps; **Analyse again** reruns it. It adds to the report and never replaces it. Each analysis uses AI credits from the organisation's allowance (chapter 22), and DilliC warns when fewer than 25 are left. Where no AI model is connected, a built-in summariser answers, and nothing is charged.

PART TWO

Administration

For account administrators: shaping DilliC to the organisation, from its hierarchy and people to its workflows, connections, and bill. Everything here is in the **Administration** group at the foot of the left pane, in five groups of its own that open when clicked; the one holding the page you are on opens by itself.

Few of these tasks need a specialist, and none needs a programmer except a receiver for signed webhooks and a pipeline that reports deployments. The table says who usually does each, and what to have at hand before starting.

CHAPTERS	WHO USUALLY DOES IT	HAVE AT HAND
12 to 14: modules, hierarchy, people and roles	An administrator who knows how the organisation is shaped	An organisation chart, and each person's name, email and place in it
15 and 16: vocabulary, estimation, priority, fields	An administrator, with the team leads	The words and scales the teams already use
17 and 18: workflows, approvals, automation	An administrator, with whoever owns the way work is delivered	The statuses the teams use today, and who must sign off what
19: measures/KPIs and scaling	An administrator, with whoever reports the organisation's figures	The measures already counted, with their units and ranges
20 and Appendix E: identity and provisioning	An administrator, sitting with whoever runs the directory (Okta, Entra ID or Google Workspace)	Administrator access to the directory, and somebody who can add a DNS record for the domain
21: connections	An administrator, with whoever administers the other tool; a developer for a pipeline or a signed webhook	Administrator rights in that tool, and for a tracker an API token from it
22: billing	Whoever pays	A card, the invoice address and the tax number
23: data and feedback	Any administrator	Nothing

CHAPTER 12

Administration and modules

Who can administer

Administration is open to **account administrators** only: people with "Administers the account" ticked (chapter 14). Anyone else who reaches it is told to ask an administrator. Every administration page is headed with the section's name, the organisation, its plan, and how many seats it holds.

> **MOST CHANGES TAKE EFFECT AT ONCE**
> Many controls here save as soon as you change them, and several removals happen without asking again. Where a removal would lose something, DilliC refuses it and says what is in the way instead: a status holding work, a role still assigned, a KPI still in use. Read the refusal; it names the next step.

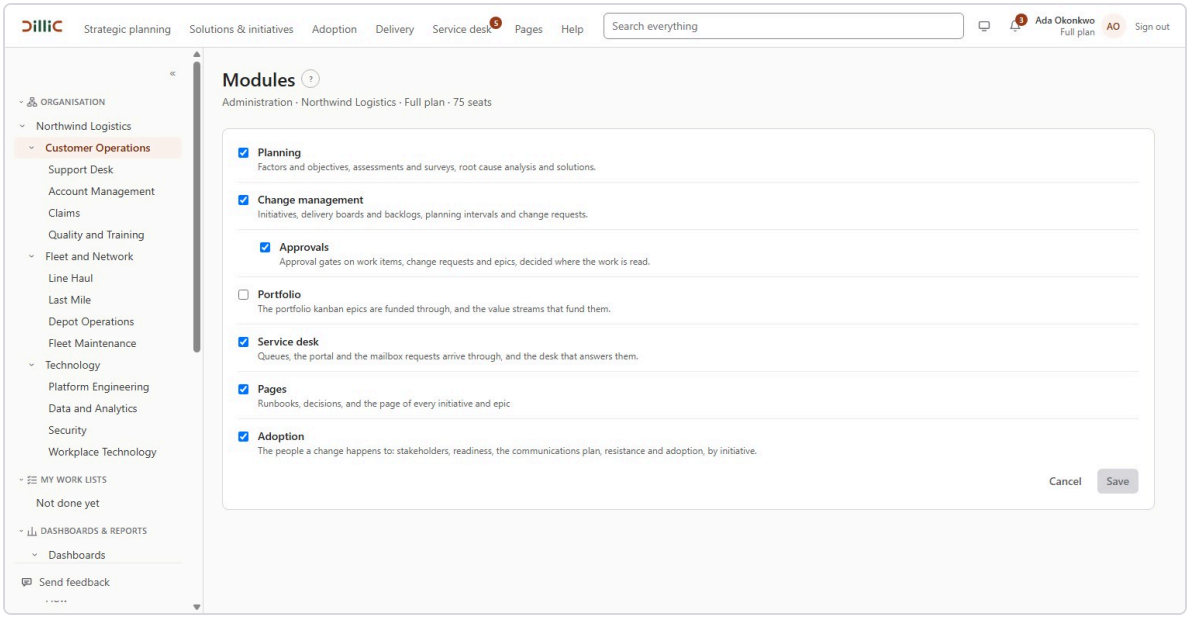
The sections

The sections are drawn in five groups, by what they belong to. A module switched off takes its group's sections away with it, so the list says what is running without a sentence.

GROUP	SECTION	FOR	SHOWN WHEN	CHAPTER
The organisation	Modules	Which parts of DilliC run	Always	12
	Hierarchy	Levels and entities	Always	13
	People	Accounts and who holds which role where	Always	14
	Roles	What each role may do	Always	14
	Identity & provisioning	Single sign-on, SCIM, two-factor policy	Always	20
	Vocabulary & estimation	The organisation's words, currency, scales and sizes	Always	15
Change management	Workflow & approvals	Statuses, moves, approval gates, risk	Change management	17
	Environments	The places work runs, in order, and their authorisation	Change management	21
	Automation	Rules that act on work	Change management	18
	Priority	How work is ordered	Change management	16
	Components, labels & links	Ways of organising work	Change management	16
	Fields	Questions of the organisation's own on work	Change management	16
	Scaling	How much of SAFe runs	Change management	19
Planning	Measures/KPIs	The measure/KPI library	Planning	19
	Groups	Who a change happens to and who a survey goes to	Planning or Adoption	14
Connections	Version control & CI	Repositories and pipelines in	Change management	21
	Outbound webhooks	What happened here, posted out	Always	21
	Apps	Third-party apps given a seat	Always	21

	Two-way sync	A board kept in step with another tracker	Change management	21
Account	Data	Where data lives; exporting it; a board in or out	Always	23
	Billing	Plan, seats, credits, invoices	Always	22
	Feedback	What people have reported	Always	23

Modules



Modules: which parts of DilliC the organisation runs.

A module is on or off for everybody, administrators included.

MODULE	TURNS ON
Planning	Factors and objectives, assessments and surveys, root cause analysis, and solutions.
Change management	Initiatives, delivery boards and backlogs, planning intervals, and change requests.
Approvals	Approval gates on work items, change requests, and epics. Needs Change management.
Portfolio	The portfolio kanban that epics are funded through, and the value streams that fund them. Stands on its own.
Service desk	Queues, the portal and the mailbox requests arrive through, and the desk that answers them: the Service desk tab in the header, the Service desk report, and the right to answer requests. Stands on its own, so a team can answer requests without running its boards here.
Pages	The wiki: runbooks, decisions, and the page of every initiative and epic. Needs neither of the two halves, so an organisation that only plans keeps its pages.
Adoption	The people a change happens to: stakeholders, readiness rounds, the communications and training plan, the resistance log, adoption targets, and the saturation heatmap, by initiative. Hangs off initiatives, which both halves have.

1. Tick or untick the modules. Approvals, Portfolio, and Service desk grey out while Change management is off.
2. **Save**.

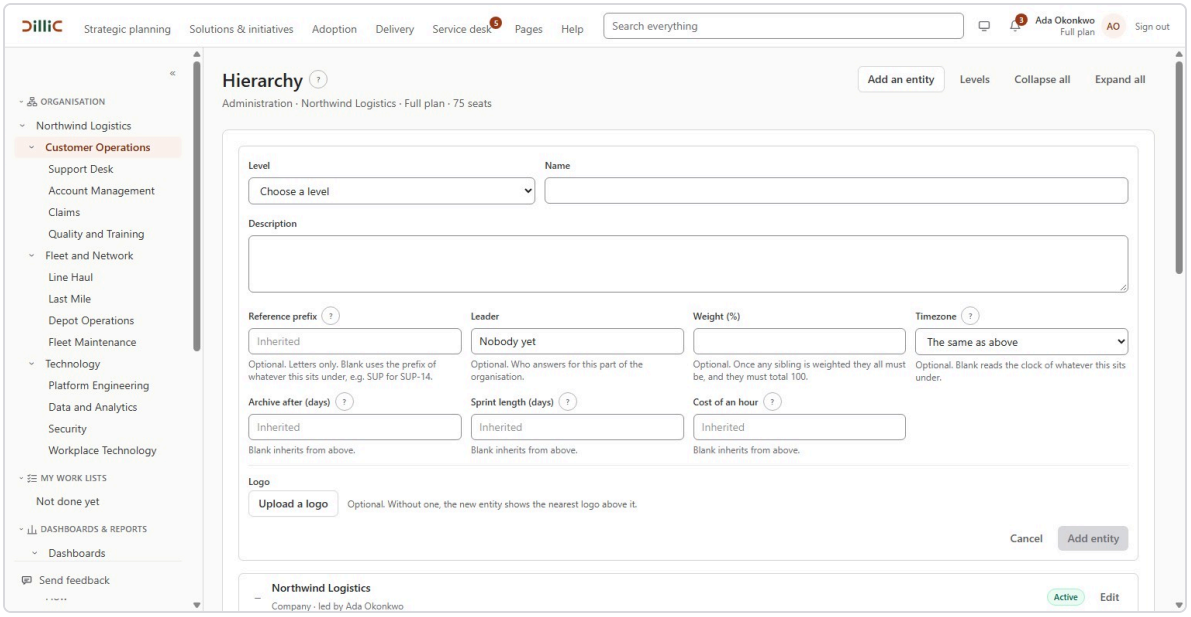
At least one of Planning and Change management must stay on; solutions and initiatives are available with either. Switching a module off hides its screens, its administration sections, its reports, notifications and gadgets, and the rights roles hold in it. It deletes nothing: switch it back on and everything is where it was. With Approvals off, gates stay configured but no longer hold anything up.

A new organisation's first administrator lands here first.

CHAPTER 13

Hierarchy

The hierarchy is the shape of the organisation: a chain of **levels** (Company → Division → Team) and the **entities** in each. Everything in DilliC belongs to an entity, and roles reach down it.



The hierarchy: the tree of entities, with Add an entity open across the top.

Across from the **Hierarchy** heading, **Add an entity** and **Levels** each open a panel the full width of the page, above the tree of entities; **Collapse all** and **Expand all** fold the tree. The (?) beside the heading explains the tree and plays the how-to.

Levels

1. **Levels**, then **New level**. Name the top level (a company, a directorate, whatever the organisation calls itself) and leave it at "Top of the hierarchy". **Add level**.
2. Add each level beneath. **Sits under** defaults to the deepest existing level.

Levels form a chain: one top level, and each other level beneath exactly one. Renaming a level is always safe. A level can be deleted only when it has no entities and no level beneath it.

Entities

In **Add an entity**, the fields sit in rows - its level, parent and name; its description; how its work is numbered, who leads it, how it counts, and what clock it keeps; how it is run - with a **Logo** to choose beneath them, sent once the entity exists:

Level	Which level it is.
Parent	Which entity it sits under. DilliC offers only entities of the level above.
Name, Description	What people read.
Leader	Optional. Who answers for this part of the organisation.
Timezone	Optional. The clock change windows raised here are agreed in; blank reads the zone of whatever it sits under.
Weight (%)	Optional. How much it counts in a composite assessment (below).
Archive after (days)	How long finished things (reviewed changes, completed or cancelled initiatives, superseded assessments) wait before archiving themselves. Blank inherits; the default is two weeks.
Sprint length (days)	The default length when someone plans a sprint here. Blank inherits.
Reference prefix	What work raised here is numbered in, e.g. SUP for SUP-14. Letters only; blank inherits, and the top of the tree uses the organisation's own, taken from its name. Changing it affects only items raised afterwards - everything already numbered keeps its reference.
Cost of an hour	Optional. When set, labour on an initiative is worked out from the hours logged, and shown as derived.

Add entity. The tree beneath lists every entity with its level, weight, leader, and time zone. **Edit** opens the same fields in the same rows, with **Status** first in the last row, and changes any of the above, including where an entity sits; moving an entity moves everything beneath it.

Logo. Under an entity's fields, **Upload a logo** gives it a mark of its own: a PNG, a JPEG, a GIF, or a WebP up to a megabyte. An entity without one shows the nearest logo above it, and the control says whose it is showing, so a company sets one and every team beneath carries it until a division with a mark of its own uploads it. The logo is shown in the header for whichever entity is chosen, at the head of its reports, on its service desk's portal, and in the letters the desk and a survey send to people outside. **Remove the logo** goes back to inheriting.

An entity with nothing beneath it runs its own delivery board; one with entities beneath reads theirs rolled together. Adding or moving entities rearranges the boards accordingly.

Weights

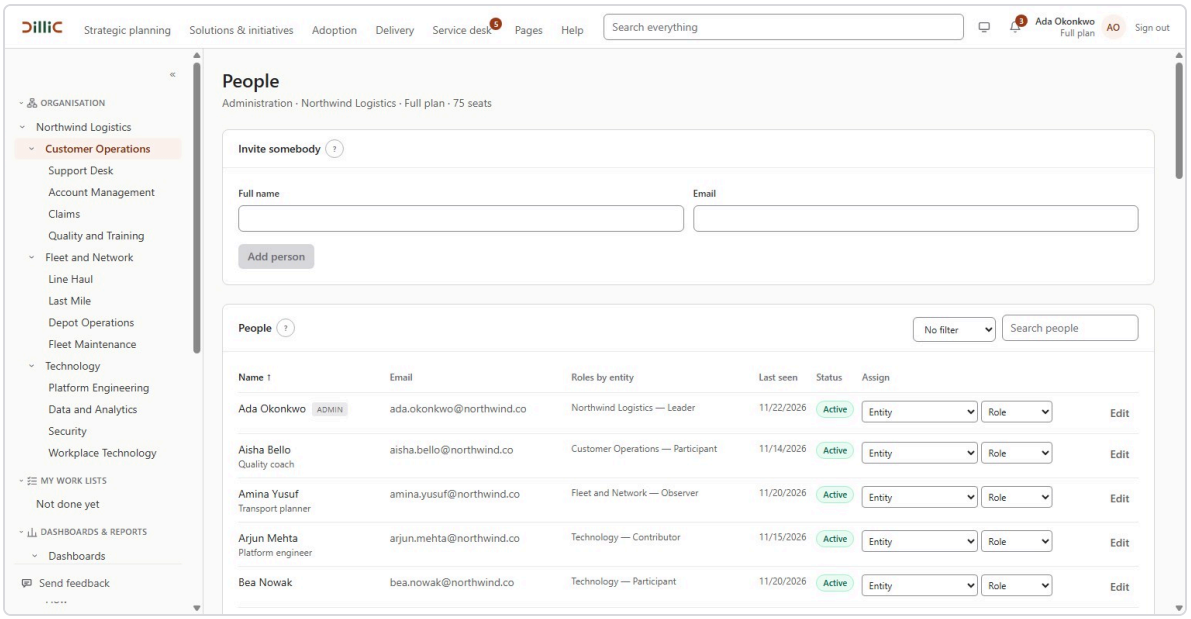
Weights are optional, and apply among siblings: the children of one parent. Once any sibling is weighted, all of them must be, and their weights must total 100. The form shows the total as you type ("Siblings would total 80% — must be 100%"). Weights decide how much each child's assessment counts in a composite at the parent; unweighted children count equally.

Archiving and deleting

To retire an entity, **Edit** it and set **Status** to Archived. It keeps its history and is no longer offered for new work. Only an empty entity can be deleted, and DilliC asks first: one with entities beneath it, or holding assessments, work, or anything else, must be archived instead, and an organisation always keeps at least one entity.

CHAPTER 14

People and roles



People: adding someone, and everyone with the roles they hold.

Adding people

1. Under **Invite somebody**, enter their **Full name** and **Email**, and their **Title** if you know it ("Sales manager").
2. **Add person**. DilliC emails them a link to choose their own password, and says the email is on its way.
3. Give them a role (below).

Nobody types a password for them. The link is good for three days; once it has run out, they ask for another from the sign-in page. Somebody added as inactive gets no email until they are reactivated. Each active person takes a seat: when every seat is taken, add seats under Billing or deactivate someone. People can have any email address, on any plan. An email address can belong to only one person in DilliC.

> **LETTING THE DIRECTORY DO IT**
>
> With single sign-on or SCIM (chapter 20), the organisation's directory can create accounts instead: on first sign-in, or in advance when someone joins.

Giving someone a role

In the person's row, under **Assign**, choose an **Entity**, then a **Role**. It saves as you choose. A person holds one role per entity: choosing another on the same entity replaces it. The role reaches everything beneath the entity, so give it at the highest entity it should cover.

People can be searched (every word must match their name, email, title, status, "admin", or the entities and roles they hold) and filtered by status or role; it shows fifty at a time. Each row shows the roles held by entity, when the person was last seen, and their status.

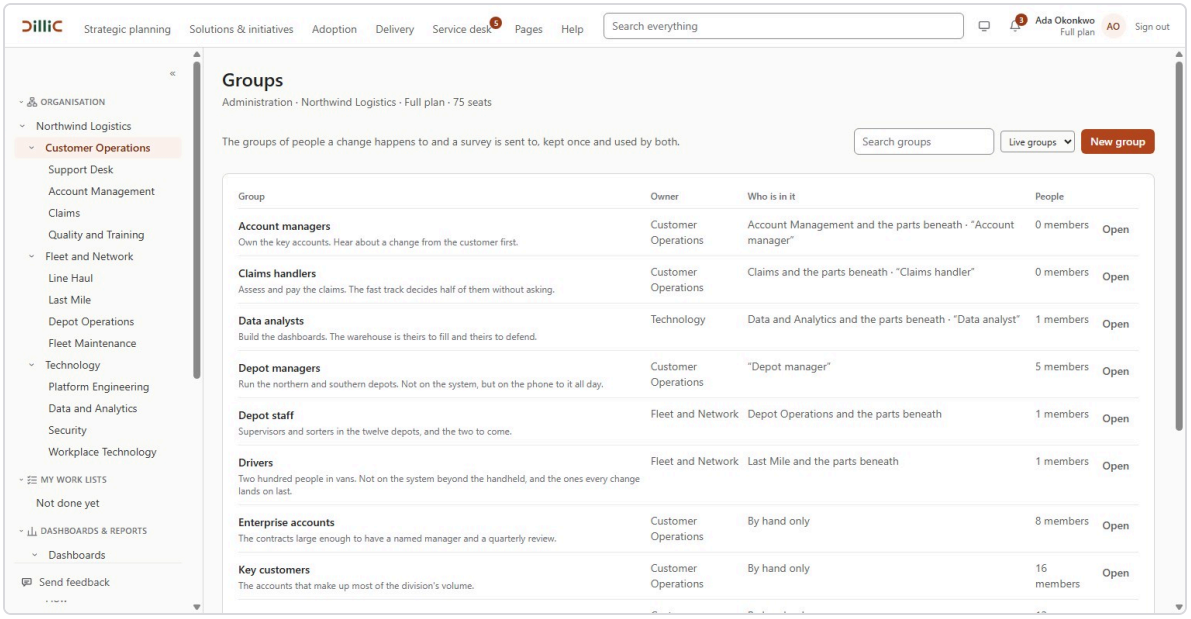
Editing and deactivating

Edit on a person changes:

- their **Full name** and **Email**;
- their **Title**, the job they do. Where the directory sends titles (chapter 20), it is filled in from there, marked **From the directory**, and changed there rather than here. A group can take in everybody with one title.
- their **Status**. **Inactive** stops them signing in and frees their seat; everything they wrote stays. Reactivating needs a free seat.
- a **New password**, or blank to keep theirs;
- **Administers the account**. The organisation must keep at least one administrator.
- **Roles held**: × removes a role at once.

Anonymise, in the same panel, is for someone who has left and asked to be forgotten. It asks again before acting: their name, email address, picture, and second factor are removed everywhere, the account can never be signed into again, and what they wrote stays, credited to a former member. It cannot be undone. You cannot anonymise yourself, or the only administrator.

Groups



Groups: who a change happens to and who a survey goes to, kept once.

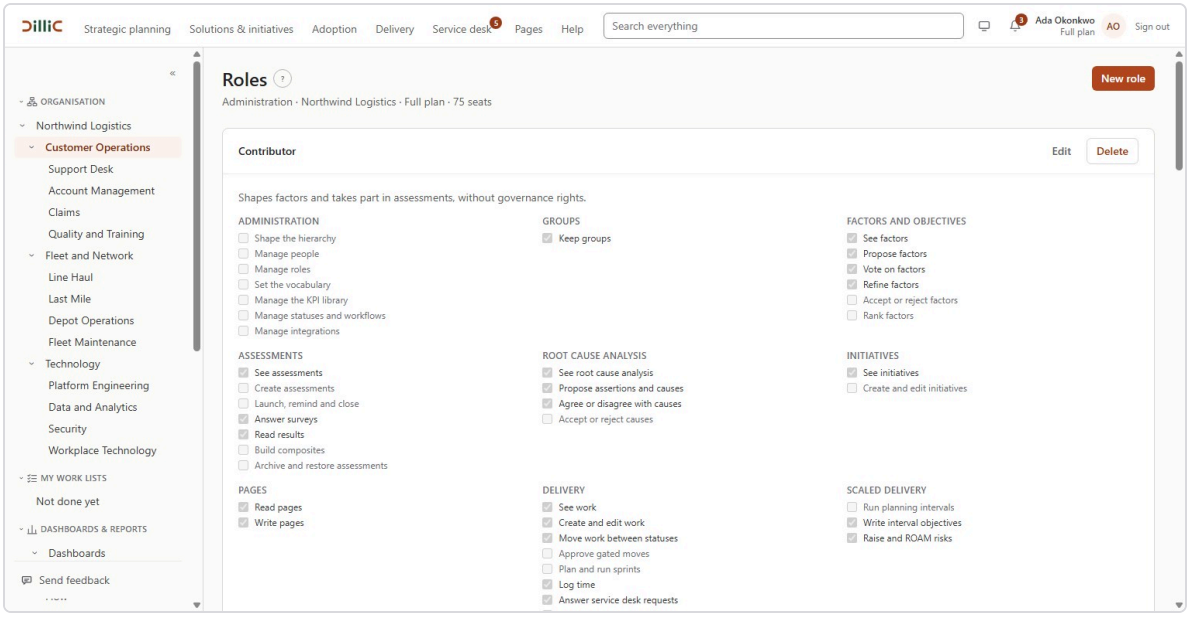
Groups is one library of the people a change happens to and a survey is sent to. Adoption places them on an initiative's stakeholder grid; an assessment invites the people outside the organisation in them. A person can be in as many groups as fit.

1. **New group**. Give it a **Name** ("Sales managers"), an **Owner** - the part of the organisation it belongs to; whoever may keep groups there may change it - and, if you like, a description.
2. Under **Who is in it**, set a rule: a **Part of the organisation** (and the parts beneath), a **Role**, a **Title**, or any of them together. Everybody the rule takes in is a member, and stays one as people join, move, and leave.
3. Add anybody else by hand: **Add a person** for somebody with a DilliC account, or **Add by email** for people outside the organisation, one per line or separated by commas (`Dana Ruiz <dana@example.com>` works too). An address that belongs to somebody in DilliC adds them as themselves.

The member list says how each person came in - **By rule**, **Added**, or both - and marks anyone **Inactive** or **Unsubscribed**. Somebody in by the rule leaves only when the rule changes. People outside the organisation are kept as contacts: they can unsubscribe or ask to be forgotten, and forgetting takes them out of every group.

Groups needs Planning or Adoption to be running. Keeping them takes the right **Keep groups** on the group's owner, which leaders and contributors have; somebody who holds it without administering the account sees only this tab under Administration. Others who pick a group - placing it in Adoption, or inviting it to a survey - see its name, its rule, and how many are in it, never who.

Writing a role



Roles: each a named set of rights, grouped by area.

Roles lists each role with its description and rights. The four shipped roles (chapter 3) are a starting point: edit them, or write your own.

1. **New role**. Give it a **Name** ("Release manager") and a description of what it is for.

2. Tick its rights, grouped by area: Administration, Factors and objectives, Assessments, Root cause analysis, Initiatives, Pages, Delivery, Scaled delivery, Portfolio, Change enablement, and Reporting.
3. **Create role**. Then give it to people under People.

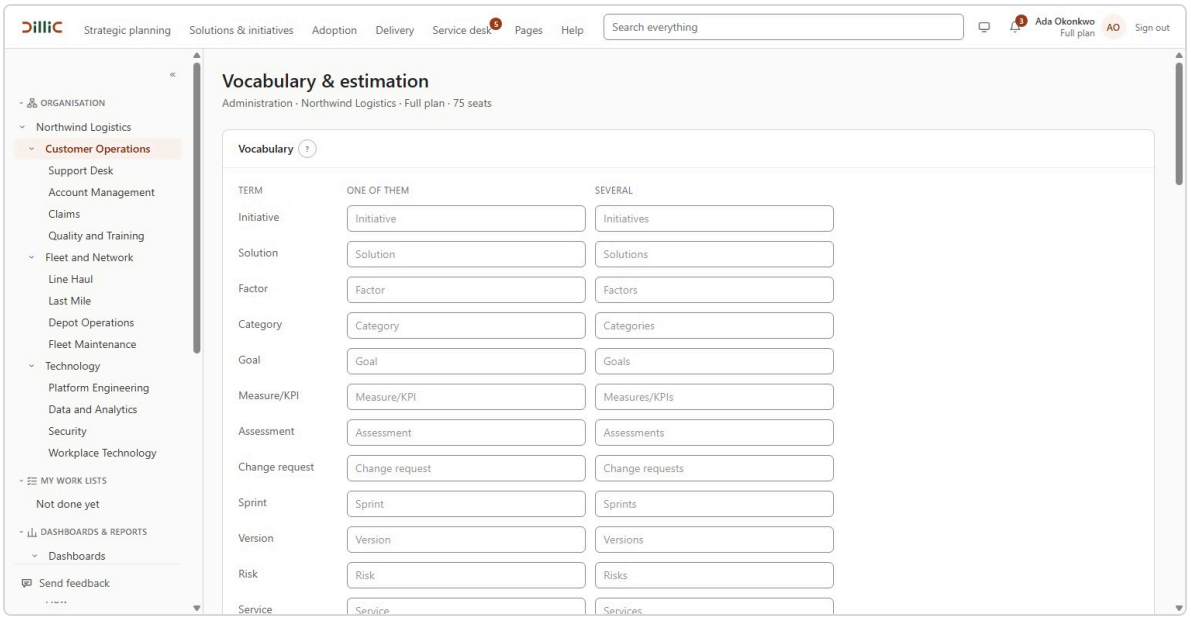
DilliC offers only the areas of modules the organisation runs; a role keeps any rights it already holds in a module that is switched off. A role still held by anyone cannot be deleted.

Rights worth knowing by name:

workflows	Configure boards: columns, cards, quick filters, methodology.
extend	Add statuses of an entity's own, where the organisation allows it.
approve	Decide approval gates that name no role (work, or change requests).
manage (sprint)	Plan and run sprints, versions, and service desk queues.
accept	Accept or reject factors, and decide root cause analysis.
launch	Launch, remind, and close assessments, and invite people outside.

CHAPTER 15

Vocabulary and estimation



Vocabulary: the organisation's own words for DilliC's nouns.

Renaming the nouns

Vocabulary renames DilliC's nouns to the organisation's own. For each, give the word for **One of them** and **Several**; it saves when you click away. An empty box shows the shipped word; clearing a box goes back to it. If only the singular is given, the plural adds an "s".

The nouns are initiative, solution, and measure/KPI; factor, category, goal, and assessment (Planning); change request, sprint, version, risk, and service (Change management); value stream (Portfolio). While SAFe runs, planning interval and PI objective are listed and locked as SAFe's own words. Renaming changes only what people read: exports and integrations keep the original names. Every (?) on a screen speaks the renamed nouns too, so an organisation that calls sprints iterations never reads "sprint" in an explanation.

The breakdown of work

The breakdown renames the rungs of work (epic, story, task, defect, sub-task), and adds rungs of your own: **Add a kind of work**, name it ("Spike"), and choose what it **Behaves like**. A new rung copies that rung's depth, estimate unit, board behaviour, and prioritisation. Rungs you added can be removed once no work is filed under them; shipped rungs stay, and are made yours by renaming. While SAFe runs, its rungs are locked.

Survey scales

Every assessment is answered on three organisation-wide scales from −3 to +3: **Strength / weakness**, **Opportunity / threat**, and **Agreement** (used by factor votes and root cause analysis). Here you can:

- **How a survey is taken**: **Two-dimensional** asks both questions of every factor; **Traditional** asks which of the four a factor is and then how much, with no middle. It saves at once. The traditional way greys out the middle box of the two survey scales, and Offer Neutral then speaks for the agreement scale alone.
- **Offer Neutral**: seven options with a middle, or six, where everybody leans one way. It applies to all three scales (to the agreement scale alone the traditional way) and saves at once.
- Reword each scale's seven labels, weakest or most negative first, then **Save**. A preview shows how a question will look.

Rewording changes how every assessment reads, past ones included, but never changes a score.

Matrix quadrant text

This is the faint line drawn behind each quadrant of every matrix. Four boxes, up to 100 characters each; **Use default** restores the shipped line, and an empty box shows nothing in that quadrant.

Estimation and sizes

Estimation chooses, for each rung that is estimated, whether it is sized in **Points** (added up on a burndown and against a sprint's capacity) or **T-shirt sizes**. Epics ship on sizes; points are for rungs planned into sprints.

> **SWITCHING A RUNG'S UNIT CLEARS ITS ESTIMATES**
> There is no honest conversion between points and sizes, so switching a rung's unit removes the estimates written in the old one, and DilliC says how many items lost theirs. It happens as soon as you choose, without asking again.

Sizes edits the T-shirt ladder: each size's label and rank (smaller first), and **Points up to**, which decides how a number imported from another tool maps onto a size. The largest size needs no threshold. Sizes are never added up.

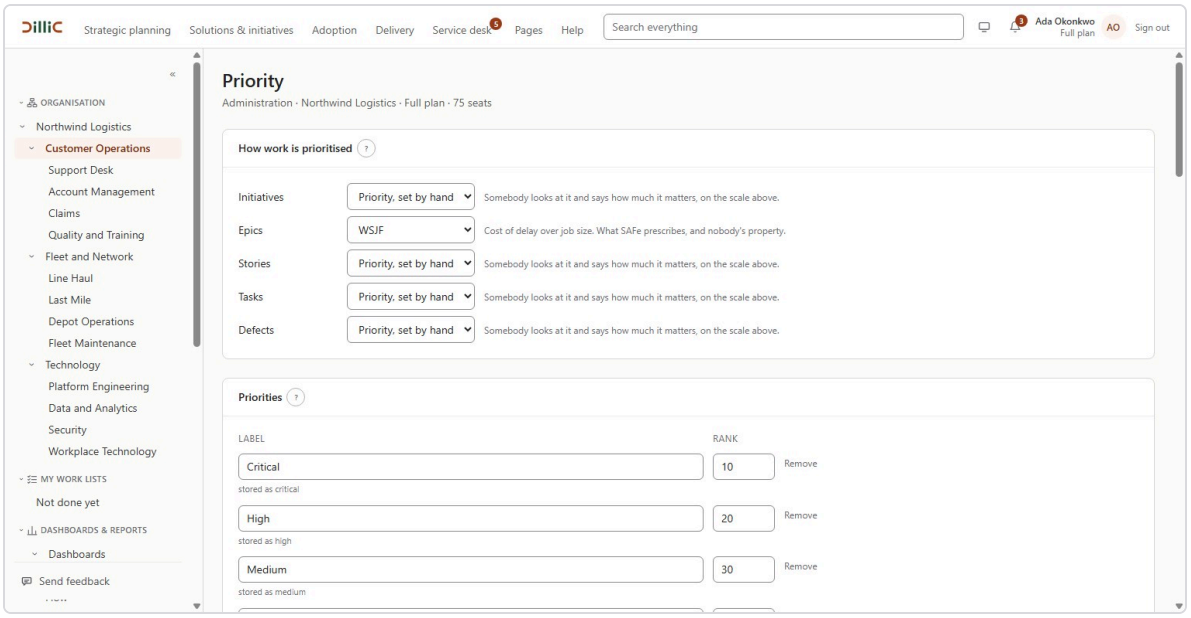
Currency

Budgets are kept in a three-letter currency (USD by default), which new budgets start in. Changing it converts nothing. A value stream may use its own currency, and initiatives drawing on it must match; DilliC never converts between currencies.

CHAPTER 16

Priority, components, labels and fields

How work is prioritised



Priority: a method for each level of work, and the scales the methods read.

For initiatives (one answer for the whole organisation) and for each rung of work, choose how it is ordered:

METHOD	IN SHORT
Priority, set by hand	Somebody says how much it matters, on the priority scale.
RICE	Reach × impact × confidence, over effort. The only one that asks how many people.
ICE	Impact × confidence × ease. RICE with the counting taken out.
WSJF	Cost of delay over job size. What SAFe prescribes.

It saves as you choose. Switching method clears nothing: only the current method's numbers are shown, and old ones are cleared when an item is next scored. Sub-tasks are not prioritised.

Priority and scoring scales

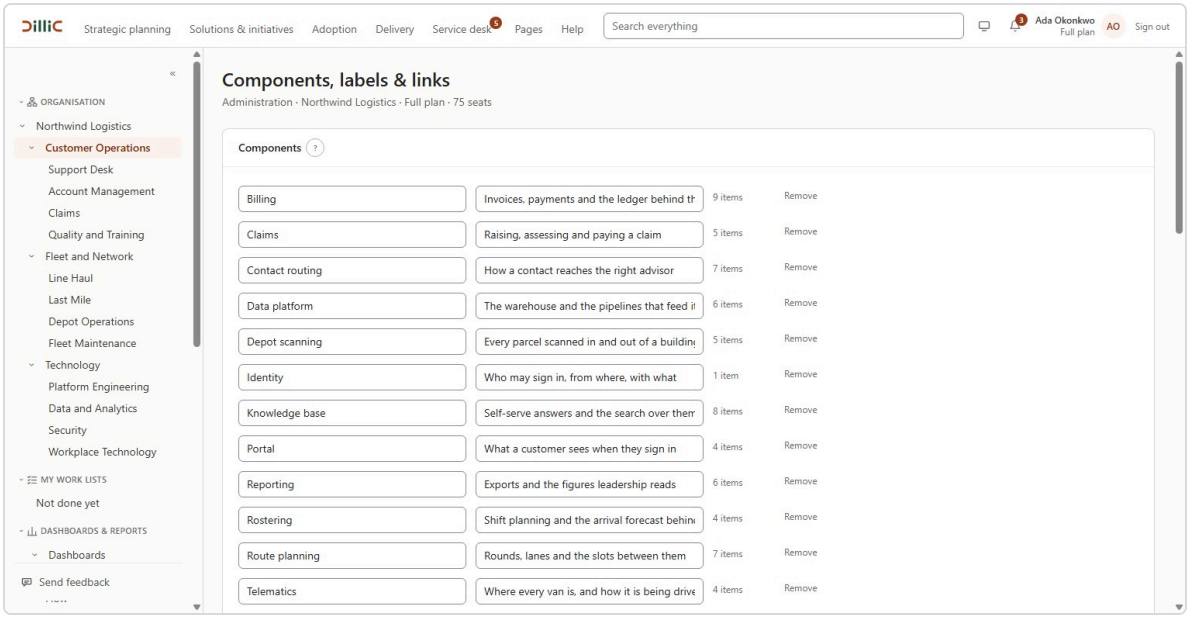
Below, each method's scale can be edited: **Priorities** (a lower rank is more urgent), **RICE impact**, **RICE confidence**, **ICE**, and **WSJF**. Each option has a label, a rank (its position; ranks must differ), and, on the scoring scales, a **weight**: the number it stands for in the sum.

1. Rename labels, adjust ranks and weights, or **Add option**.

2 **Save scale**, or **Discard changes**.

Renaming an option leaves the work on it alone: the label is what people read, and the key beneath it (shown as "stored as") is what the data is kept against. An option still in use cannot be removed; rename it instead. Weights must be above zero.

Components and labels



Components, labels, and link reasons.

- **Components** name the parts of the system work touches. The list is closed: only administrators add to it. Give each a name and what it means; a rename reaches every item carrying it, and removing one takes it off its items.
- **Labels** are open: anyone creates one by typing it on an item, ignoring capitals. Administrators tidy them here. Two labels cannot share a name, so merge a stray spelling by removing it.

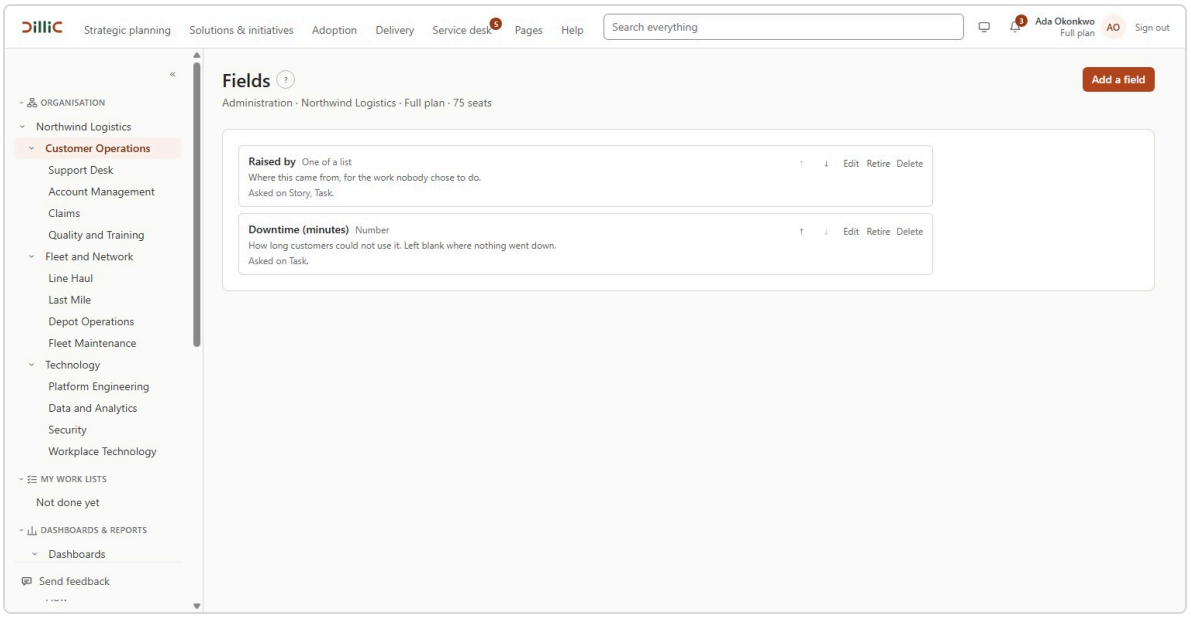
Link reasons

The reasons two items can be linked sideways, each worded from both ends: "Blocks" / "Is blocked by". **Add a reason**, write both wordings, and **Save reasons**. A reason still holding links cannot be removed; rename it instead. There must always be at least one.

Requests

People who file work can ask for a missing component or link reason from the item. Each request shows who asked, where, and why. Adjust the wording and **Add it**, or write a sentence and **Decline**. Either way they are told.

Custom fields



Fields: the organisation's own questions on work.

A custom field behaves like a built-in one: it can be a backlog column, a condition in a list or report, and a column in an export.

1. **Add a field**. Write the **Question** ("Regulator").
2. Choose its **Kind** (below); it cannot be changed later.
3. Say **What it means**, shown beside the box.
4. For list kinds, write the options.
5. Under **Asked on**, tick the rungs it is asked on, and for each whether it **Has to be answered**.
6. **Add the field**.

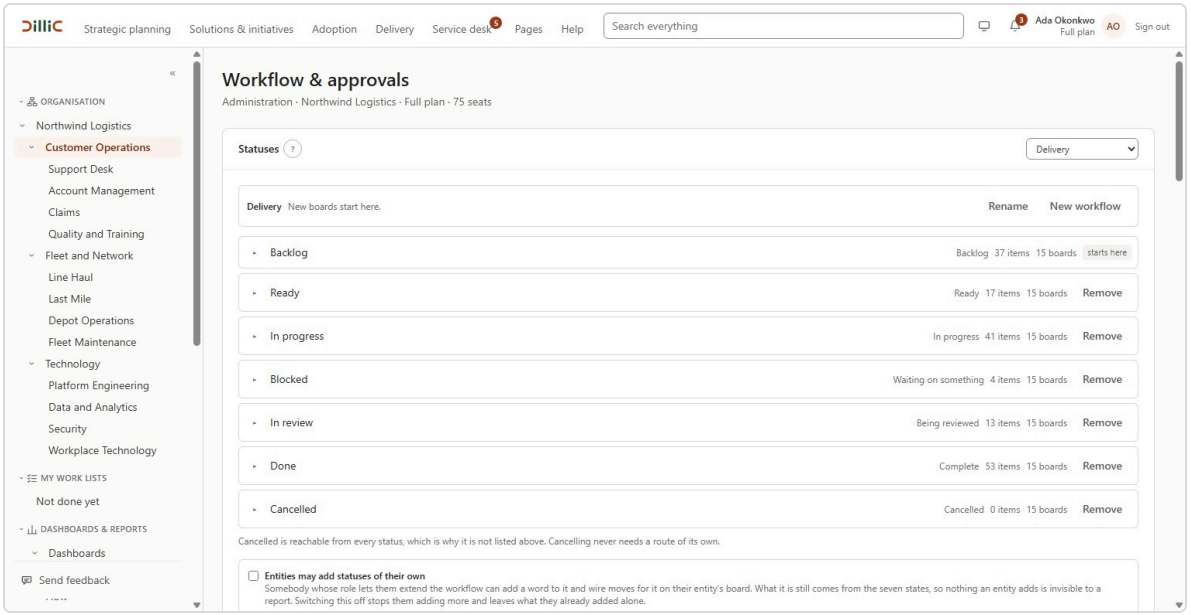
KIND	HOLDS
Text	One line: a reference in another system, a name.
Long text	Several lines, in Markdown.
Number	A quantity reports can total and average.
Date	A day.
One of a list	One answer from a list you write.
Any of a list	Any number of answers from the list.
Person	Somebody in the organisation.
Yes or no	A box; unticked is an answer.
Link	A web address.

↑ ↓ reorder fields. **Retire** stops a question being asked and keeps every answer; **Ask again** brings it back. **Delete** asks first, because it removes every answer ever given. To change a field's kind, add a new field and retire the old one. An option in use cannot be removed; rename it instead.

CHAPTER 17

Statuses, workflows and approvals

A workflow is the set of statuses work moves through, the moves allowed between them, and the approvals some moves need. It is configuration, not code, and it lives under **Workflow & approvals**.



Statuses: each with the state it means, the work in it, and the boards that show it.

Statuses and states

Choose a workflow in the menu at the top. Each of its statuses shows the **state** it means, how many items are in it, and how many boards show it; one is marked *starts here*, where new work arrives. Open a status (▸) to rename it, choose its state, and tick where it can go next.

Every status is one of seven states: Backlog, Ready, In progress, Waiting on something, Being reviewed, Complete, and Cancelled (appendix B says what each means). The state is how DilliC compares boards that name their statuses differently, draws burndowns, and counts work in reports.

Add a status: name it ("Awaiting customer response"), choose its state, tick at least one status that can lead to it, and **Add**. It is offered to every board; put it on one from the board's Configure.

DilliC keeps every workflow workable, refusing an edit that would break it:

- at least one status must mean Complete;
- every status other than the first must be reachable by some move;
- every status that is not Complete or Cancelled must have a way out;
- the status where work starts cannot be Complete or Cancelled;

- a status cannot be removed while it holds work, starts work, or is on a board.

Moves

Under **What can come after it**, tick each status a card may move to from this one. Beside each ticked move, choose an approval gate, or "no approval needed". Some moves, such as Cancel, are allowed from anywhere; they are named beneath the list and need no route of their own.

Several workflows

An organisation can run several delivery workflows: one for software teams and another for facilities, say. Reports still add up, because every status means one of the seven states.

1. Choose the workflow to start from and use **New workflow**. Name it ("Facilities"); it starts as a copy of that one's statuses, moves, and gates.
2. Edit its statuses and moves.
3. On each board that should use it, choose it in Configure → Statuses & columns.
4. **Make default** if new boards should start on it.

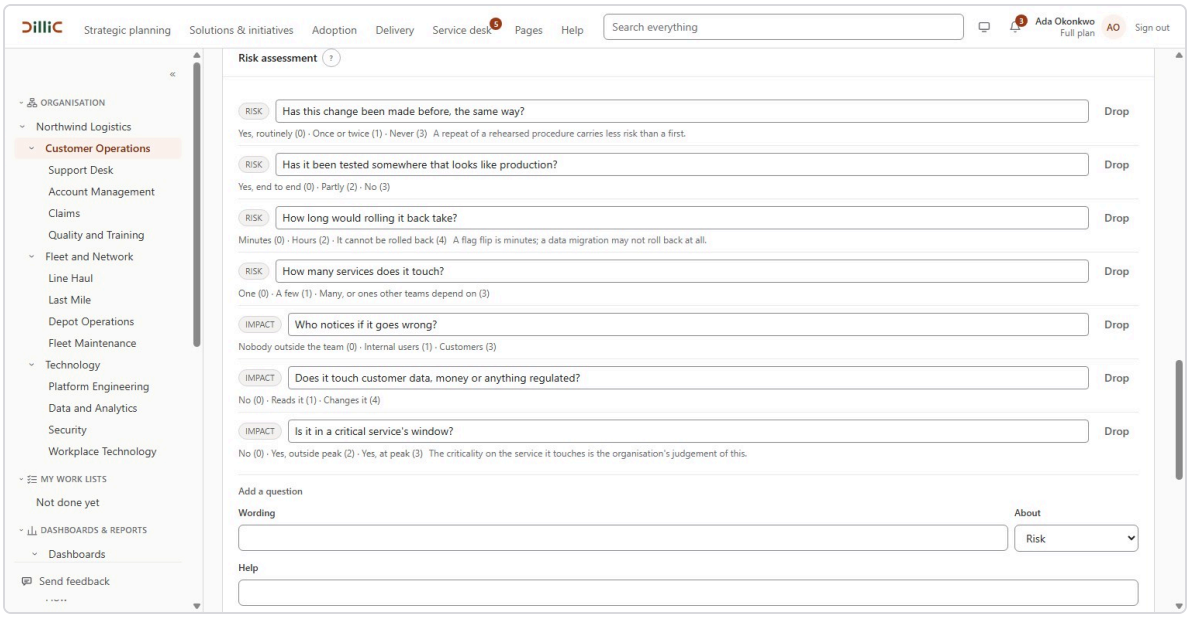
Each workflow shows how many boards use it. **Rename** is always safe. **Delete workflow** is refused while it is the default, while a board uses it, or while work sits in its statuses.

Statuses teams add

Entities may add statuses of their own lets teams add statuses to their own board, for words the organisation's workflow lacks. People also need the right to extend the workflow. A team's status still means one of the seven states, so nothing it adds is invisible to a report. Switching the setting off stops new ones and leaves existing ones alone.

Added by teams lists them. **Adopt organisation-wide** makes one the organisation's own without moving any work, unless the organisation already has a status of that name.

Risk assessment and routing



The questions a change is assessed with, and which authority each level goes through.

Under **Statuses**, **Risk assessment** is the questionnaire a change request is assessed with. Each question is **About** risk (how likely this is to go wrong) or impact (how much it matters if it does), with answers that each carry a score; write the first answer as the safe one. An account starts with seven shipped questions and may rewrite them. **Which authority, by risk** holds the routing rules: for a change of this risk - and, if named, this type - the gated authorisation move **Goes through** the gate named, or *no gate*, which applies the move on the spot. A rule naming the type beats one that leaves it open; with no matching rule the move keeps the gate the workflow put on it, so an account that writes none runs as before.

Change models

A change model is a kind of change written once and raised from many times. Under **Statuses**, **Change models** holds your library. **Write a model** and give it a **Name**, a **Type**, where it is **Offered on** (the whole organisation, or one part and everything beneath it), its implementation, rollback and test plans, the notice to the people it lands on, the services it touches, and its answers to the risk questions.

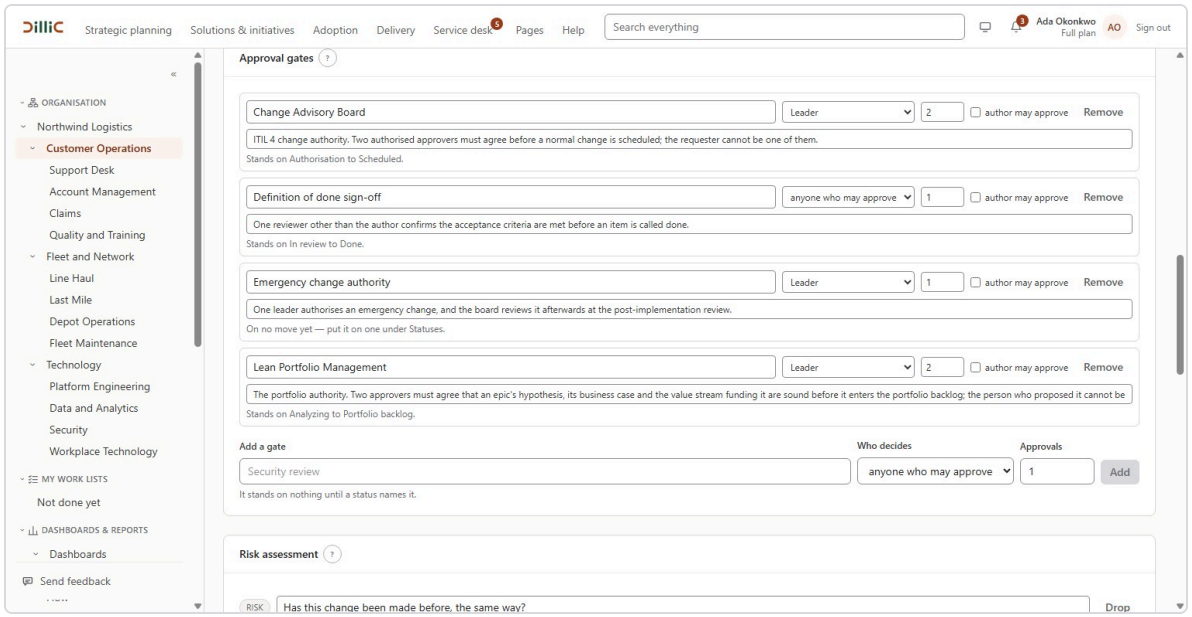
- A **standard** model is the pre-authorisation itself. ITIL authorises a standard change once, as a kind; this is where that is written down, and a change raised from the model skips the change authority.
- A change raised from a model is filled from it where the person raising it left room, and its answers score the change as an answered questionnaire would.
- **Retire it** takes a model off the raise form. A model a change was raised from is retired rather than dropped, so the change keeps saying where it came from.

Approval gates

A gate names who has to agree before a move happens. It needs the Approvals module.

1. Under **Approval gates**, name the gate ("Security review").
2. Choose **Who decides**: a role, or "anyone who may approve".
3. Set how many **Approvals** it needs, and **Add**.
4. Put it on a move: open the status the move starts from, and choose the gate beside the move.

A gate naming a role is decided by holders of that role on the item's entity or any entity above it. One naming no role is decided by whoever holds the matching right: approve work, approve change requests, or manage the portfolio. **author may approve** decides whether the person who raised the item may be one of them. Administrators may always decide. Changing a gate applies to every move it stands on and to approvals already waiting; a gate cannot be removed while it stands on a move or has approvals waiting. Sub-tasks are never gated.



Approval gates: the Change Advisory Board, a definition-of-done sign-off, and the portfolio authority.

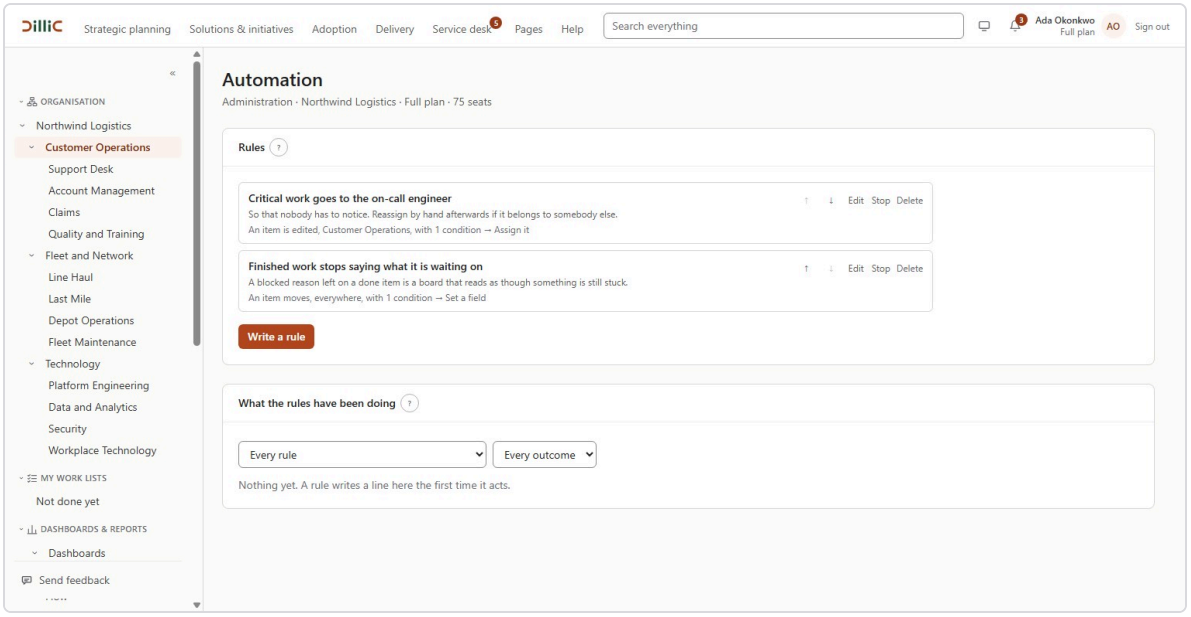
The change request workflow

Choose **Change enablement** in the workflow menu to edit the workflow change requests follow (chapter 9). Statuses, moves, and gates work the same way; there is only one, so it has no rename, default, or delete. The portfolio kanban's workflow is SAFe's own and is not edited here.

CHAPTER 18

Automation

A rule takes over something a team does by hand every week. It reads as one sentence: *when* this happens to a work item, *and* the item looks like this, *then* do these things.



Automation: the rules, and what they have been doing.

Writing a rule

1. **Write a rule**. Give it a name ("Give critical defects to the on-call engineer").
2. Choose **Where it applies**: everywhere in the organisation, or one entity. A rule on an entity applies to that entity only, not to those beneath it.
3. Say what it is for, for whoever reads it in six months.
4. Choose **When**: the trigger.
5. Add **conditions** the item must meet, or none.
6. Add one or more **actions**.
7. **Write the rule**.

Rules are listed with a summary of each. ↑ ↓ set the order in which rules matching the same event run. **Stop** turns a rule off at once, and **Start** turns it back on; **Delete** removes it, and its past runs stay in the log.

Triggers, conditions, actions

WHEN	FIRES ON
An item is created	Anything new, however it arrived: typed in, imported, split off.
An item moves	Any change of status. Narrow it with a condition on the status.
An item is edited	A field on the item changed: priority, assignee, dates.
A field this organisation added is answered	One of the custom fields was filled in or changed.
Its components or labels change	Components or labels set or changed.
An item is estimated	A size or points written on it.
An item moves in the ladder	Put under something else, or taken out from under it.
An item is archived	Put away.
Anything at all happens to an item	Every event above. Broad on purpose: narrow it with conditions.

Conditions use the same fields and tests as lists and reports (appendix A): title, type, status, state, priority, estimates, version, assignee, reporter, component, label, initiative, score, dates, and custom fields. With two or more, choose whether all must hold or any. With none, the rule runs on everything the trigger fires on, which is a legitimate and common rule.

THEN	DOES
Assign it	To a person, or to nobody.
Move it	To a status, through the workflow: a gated move asks for approval rather than moving. On a board using another workflow, it moves to the status meaning the same.
Set a field	Sets the title, priority, points, size, original or remaining estimate, due date, or a field the organisation added.
Add a component or label	One the organisation already has.
Say something	Comments on the item, in the rule's name.
Tell somebody	Notifies a person, or everybody watching the item.

Set a field offers only the fields a rule can write. The assignee is set with Assign it, and components and labels with Add a component or label.

What a rule may do

A rule runs as the person whose action set it off, and can do only what they could have done by hand: someone who cannot move work will not move it because a rule said so. It also stops itself looping:

- a rule set off by a rule set off by a rule stops at three deep;
- a rule acts on the same item only once in one chain;
- rules do not run during imports, or on their own actions.

Each successful run adds a line to the item's history saying which rule did what.

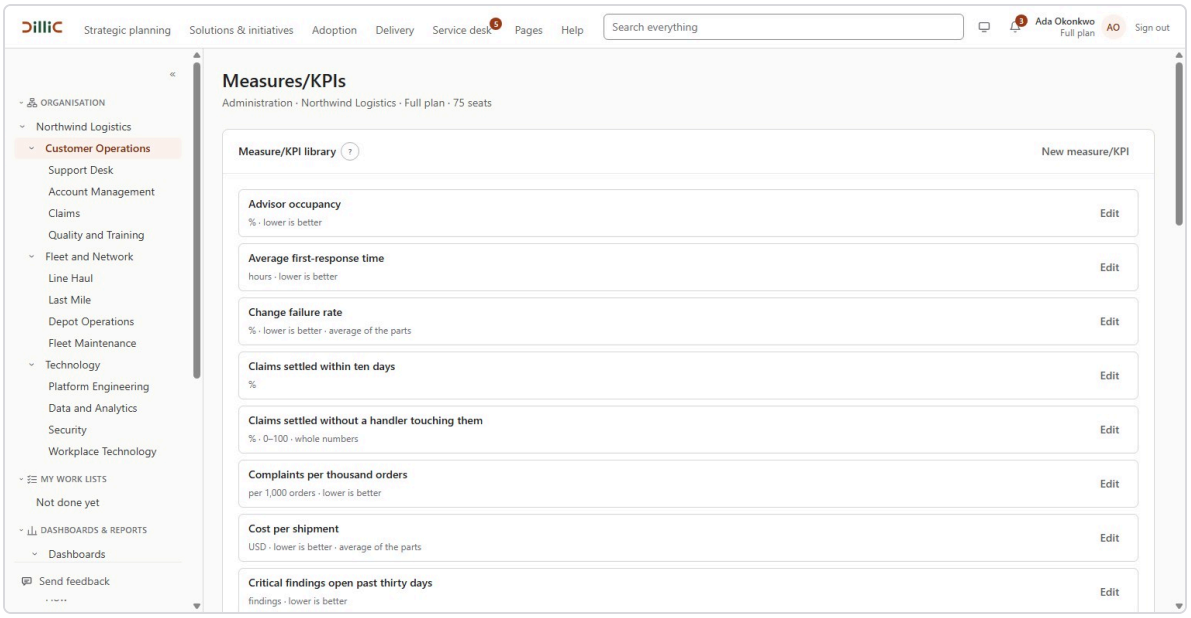
The rule log

What the rules have been doing lists every time a rule acted, or wanted to and could not, with who set it off, whether it was set off by another rule, and one of four outcomes: **Ran**, **Failed** (an action went wrong; the rest stood), **Refused** (the person could not have done it by hand), or **Stopped** (a chain ran too deep). The log leaves out rules whose conditions did not match. Anyone can read it, so people can see why something happened to their work.

CHAPTER 19

KPIs and scaling

The KPI library



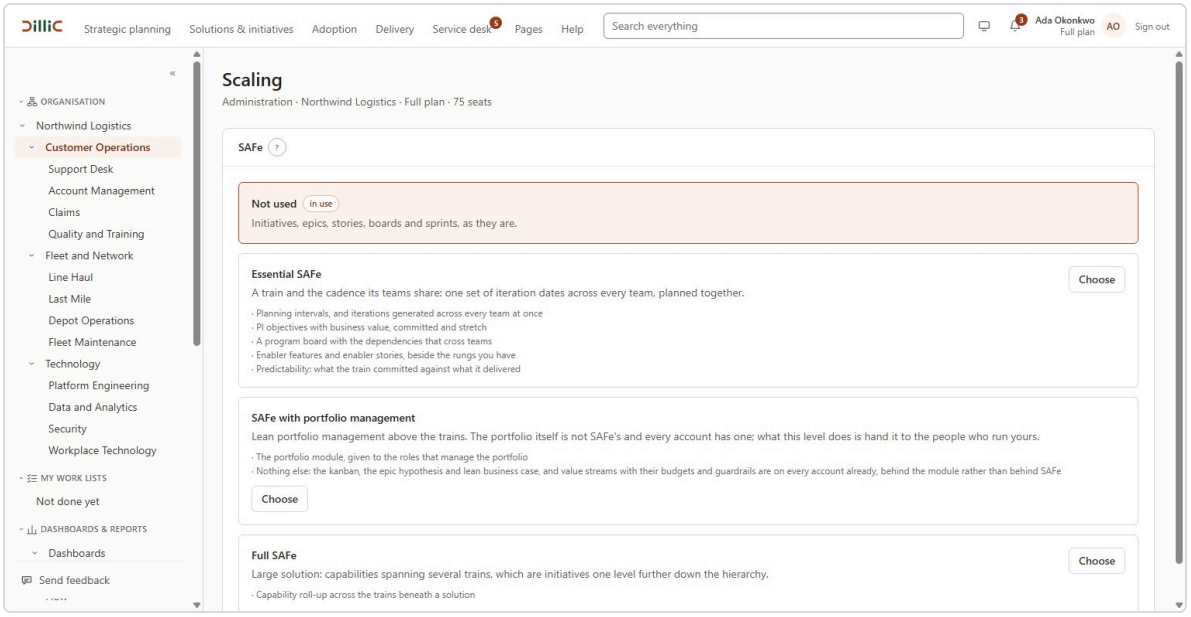
The KPI library, chosen from by factors and initiatives.

Factors, initiatives, and adoption targets choose their measure/KPI from one shared library, so each is written once, and renaming it reaches everything measured by it. The tab is called **Measures/KPIs**, a noun the organisation can rename under Vocabulary (chapter 15) to *Key results*, *Metrics*, or its own word. **New measure/KPI**:

Name, Unit	Required. The unit reads "%", "seconds", "incidents".
Low end, High end	Optional range. A reading outside it is refused.
Decimal places	0 to 6. Readings are rounded when saved.
Lower is better	For response times, defects, costs.
Rolls up as	How the KPI reads at a parent when several entities beneath measure it: does not roll up, sum (counts and money), average (rates), or weighted average (by the hierarchy's weights).
Readings come from	**Typed in**; **cycle time**, read off the work; or one of ten figures the service desk and the change log already hold: incidents, major incidents (SEV1 and SEV2), time to restore, time to first response, requests resolved on time (allowing for paused clocks), change failure rate (failed, rolled back, or caused an incident), lead time for changes (the median hours from raising a change to its end), changes implemented, availability (the share of the month with no major incident open, overlaps counted once), and requester satisfaction. Cycle time is the mean days from starting to finishing the stories finished in the month: for an initiative, its own; for a factor, every one in its part of the organisation and beneath it. Availability is read from the incidents people raise, not from a probe. The figure decides the direction, range, rounding, and roll-up. A daily job writes one reading per calendar month, twelve months back, for every factor and initiative measured by it, off the services the factor names or its part of the organisation runs, and the services the initiative establishes or enhances. Readings of such a KPI are never typed in, and a KPI with readings cannot change where they come from - make a new one instead.

A measure/KPI still measuring anything - a factor, an initiative, or an adoption target - cannot be deleted; point those at another one, or rename it.

SAFe levels



Scaling: how much of SAFe the organisation runs.

Scaling chooses how much of SAFe runs, for the whole organisation. Each level is the one before plus more:

LEVEL	ADDS
Not used	Initiatives, epics, stories, boards, and sprints, as they are.
Essential SAFe	Planning intervals with iterations shared across every team; PI objectives with business value, committed and stretch; the program board; enabler features and stories; predictability.
SAFe with portfolio management	The portfolio kanban and value streams, given to the roles that manage the portfolio.
Full SAFe	Capabilities rolling up across the trains beneath a solution.

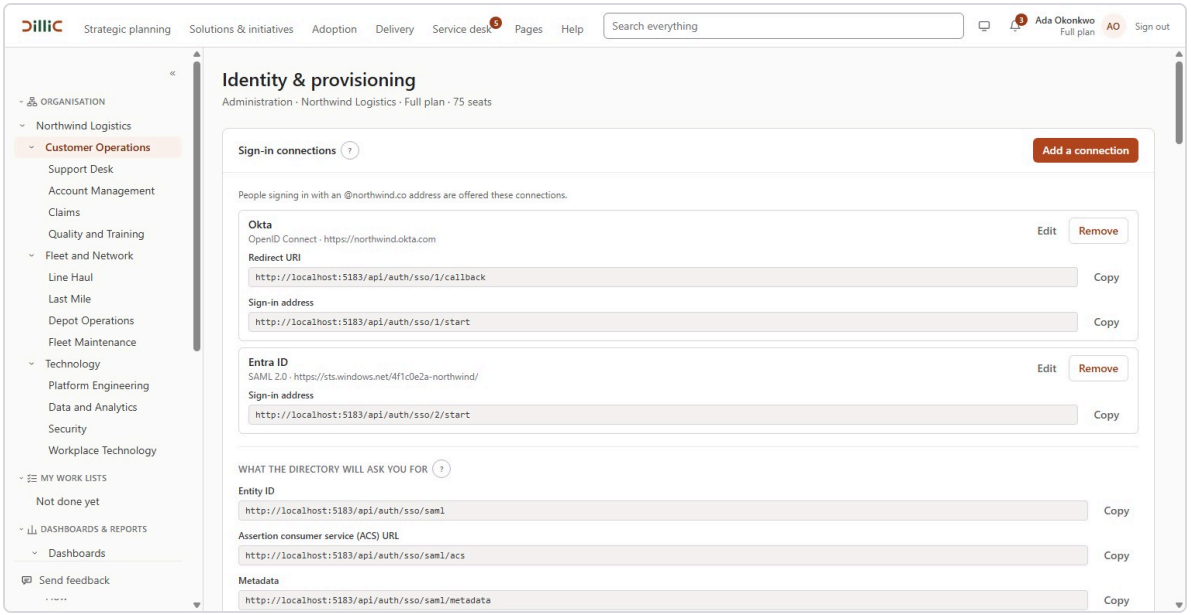
1. **Choose** a level. DilliC previews what would change and makes no change yet.
2. Read it: new rungs, sprints reading as iterations, epics reading as features and scored by WSJF, and how much work would move (raising a level re-levels and re-parents nothing).
3. **Raise the level**, or **Step back** when lowering; or **Not now**.

Lowering a level hides what it added and deletes nothing: work stays where it is, a SAFe rung holding work stays (no longer offered for new work), and raising again brings back exactly what was there. Each board keeps its own methodology throughout.

CHAPTER 20

Identity and provisioning

How people prove who they are: single sign-on through the organisation's directory, provisioning from it, and whether a password alone is enough. Nothing here grants rights; roles stay DilliC's. Appendix E gives the steps for Okta, Entra ID and Google Workspace.



Identity & provisioning: sign-in connections, the details a directory asks for, and two-factor policy.

WHO DOES THIS

An administrator, together with whoever runs the organisation's directory. Proving the domain also needs somebody who can add a DNS record. Allow an hour, with both admin consoles open side by side.

IN PLAIN WORDS

A **directory** is the system that already holds your organisation's work accounts: Okta, Microsoft Entra ID, Google Workspace. **Single sign-on** lets people into DilliC with that account, so they have no second password. **SAML 2.0** and **OpenID Connect** are the two standard ways a directory vouches for a person; they do the same job, and Appendix E says which to choose. **SCIM** is the standard by which a directory tells DilliC who has joined and who has left. A **DNS TXT record** is a line of text published under your domain's name, which only somebody who controls the domain can add. An organisation that signs in with passwords needs none of this, and can skip to Two-factor policy.

Sign-in connections

A connection is a directory the organisation trusts: Okta, Entra ID, Google Workspace. There can be several; each enabled one is a button on the sign-in page for people whose email domain is the organisation's, and the page says which domain that is.

An organisation claims the domain of the address it signed up with; following the confirmation link proved that it receives mail there. A personal mail provider such as gmail.com is never claimed, and a domain another organisation already holds stays with it. An organisation with no domain is told so here; ask the DilliC team to set one with **Send feedback**. The domain decides only who is offered the organisation's sign-in connections: people with any address can still be members.

Proving the domain

Claiming a domain proves one person receives mail there, not that the organisation runs it. So the sign-in page offers a domain's connections to nobody until the organisation proves it, the way Atlassian and Google ask for a DNS record:

1. Under **Sign-in connections**, the page shows a TXT record: its **Name** (`_dillic.` and your domain) and its **Value** (`dillic-domain-verification=` and a code of your organisation's own).
2. Whoever runs the domain's DNS adds that record.
3. **Verify**. DilliC looks the record up; once it is found, the domain is proved, and connections are offered to everybody signing in at it.

A new DNS record can take a few minutes to be seen; if Verify does not find it yet, it says what it looked for, so try again shortly. The record can be removed once the domain is proved.

OpenID Connect

1. In the directory, register DilliC as an application and note its issuer, client ID, and secret. The directory asks for a redirect URI, which DilliC shows under the connection once it is saved; Appendix E says how to break the circle.
2. **Add a connection**, protocol **OpenID Connect**. Name it; the name is what the sign-in button says.
3. Enter the **Issuer** (https; everything else is discovered from it), the **Client ID**, and the **Client secret** if there is one.
4. Check the **Scopes** (`openid email profile`) and which claims carry the email, the name, and, under **Title claim**, the job title. If it is left blank, DilliC tries `title`, `jobTitle`, and `job_title`; most directories send a title only once they are told to.
5. **Save the connection**.

Under each saved connection DilliC shows its **Sign-in address**, which starts a sign-in through it. DilliC answers only sign-ins it started, so a directory's app tile should send people there.

SAML 2.0

1. Give the directory DilliC's details from **What the directory will ask you for**: the Entity ID, the Assertion consumer service (ACS) URL, and the Metadata address. Copy them rather than typing them.
2. **Add a connection**, protocol **SAML 2.0**. Enter the identity provider's entity ID, its sign-on URL, and its signing certificate. While a certificate is being rotated, list both, separated by a blank line.
3. Check which attributes carry the email, the name, and the title, and **Save the connection**.

Creating people on first sign-in

It is off by default. When ticked, someone the directory knows and DilliC does not is created on their first sign-in, with the role and entity you choose. Leave it off where SCIM already says who exists: it is a second, quieter way for the seat count to grow. With it off, such a person is told to ask an administrator to add them.

People are matched to their DilliC account by the directory's identifier, then by email. Their name and email follow the directory at every sign-in, and so does their title when the directory sends one; a title typed by hand is kept until the directory sends its own. The client secret is never shown again once saved.

Two-factor policy

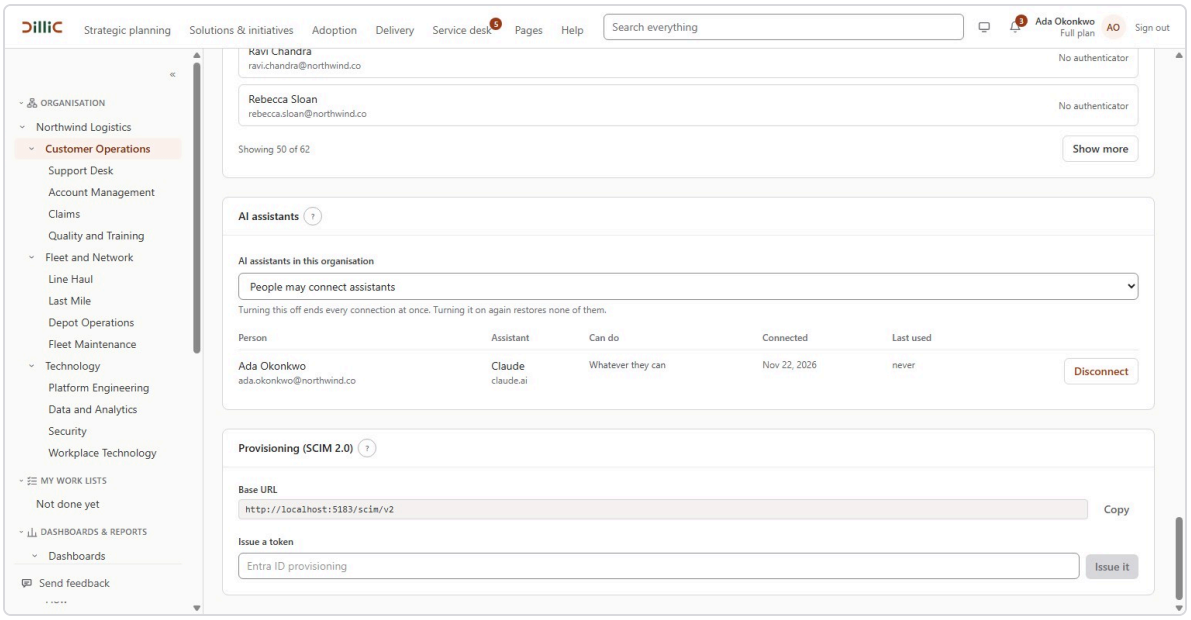
When signing in with a password, either a password is enough (though someone who has set up a second factor is always asked for it), or **a second factor is required**. Requiring it locks nobody out: anyone without an authenticator is asked to set one up at their next sign-in. It covers the password door only; people signing in through a directory rely on the directory's own checks.

A line counts the organisation: how many have an authenticator, how many sign in through the directory, and how many sign in with a password alone. The list beneath shows who has an authenticator (and how many recovery codes they have left), who signs in through the directory, and who will be asked to set one up. It is searched by name or email, narrowed with **Who is listed** to those with an authenticator, without one, through the directory, or the administrators, and shown fifty at a time with **Show more**, as People is. **Reset** clears someone's second factor when they have lost their phone; they keep their password and, where it is required, set up a new one.

AI assistants

Anyone may connect an AI assistant from their Security page, and it then acts as them. This card lists every assistant connected in the organisation: the person it acts as, the name it gave and the address its sign-in answered to, what it can do, when it connected, and when it was last used. The name is the assistant's own word; DilliC checked the address. **Disconnect** stops one at once, and its owner can connect it again.

AI assistants in this organisation is the organisation's own answer. Choose **Nobody may connect an assistant** and every connection ends at once, no new one can be approved, and the address assistants use is closed to everyone in the organisation. A script with an access token still reaches the API. Turning assistants on again restores no connection; each person connects theirs again.



Every assistant connected in the organisation, and the organisation's own answer.

SCIM provisioning

Sign-in answers "is this them"; SCIM answers "who exists, and who has left". With it, the directory creates people in DilliC when they join and deactivates them when they leave. A deactivated person keeps their history and can no longer sign in. Their job title follows the directory's `title` (Entra ID's *jobTitle*), so a group defined by title keeps itself.

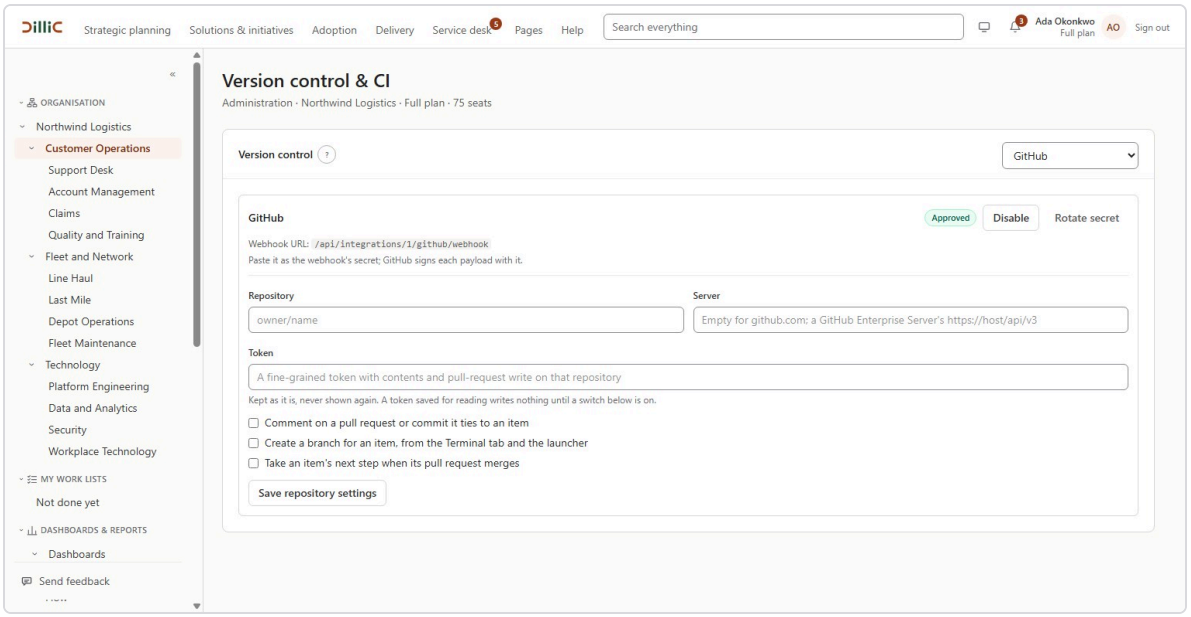
1. Under **Issue a token**, name it ("Entra ID provisioning") and **Issue it**.
2. Copy the token now: it is stored as a hash and cannot be shown again.
3. In the directory's provisioning settings, enter the **Base URL** shown here and the token.
4. To provision groups as roles, set **Where they hold it** on a sign-in connection: each directory group becomes a DilliC role held there. The field stands apart from creating people on first sign-in, which can stay off.

Tokens show when they were last used; **Revoke** stops one and keeps the record.

CHAPTER 21

Connections

Connecting DilliC to the rest of the toolchain, on four pages under **Connections** in Admin: **Version control & CI** in, **Outbound webhooks** out, **Apps** given a seat, and **Two-way sync** with other trackers. The environments the pipelines deploy to are a page of their own under Change management, explained here because the pipelines are what fill the list.



Version control & CI, the first page under Connections.

WHO DOES THIS

An administrator, with whoever administers the tool at the other end. Posting into a Slack or Teams channel and syncing with a tracker are form-filling. A pipeline that reports deployments, and a receiver of your own for signed webhooks, are a developer's work; give them this chapter and the API reference.

IN PLAIN WORDS

Version control is where developers keep code: GitHub, GitLab, Bitbucket. A **commit** is one saved change to it, and a **pull request** is a change offered for review. **CI**, continuous integration, is the service that builds, tests and deploys the code by itself; what it runs is a **pipeline**. A **webhook** is a message one system sends another over the web the moment something happens. A **secret** is a password the two systems share, so that each knows a message is genuine. An **API token** is a password made for a program rather than a person. Every secret and token is shown once: copy it into the other tool before closing the page, and if it is lost, issue a new one.

Version control and CI

Choose the service: GitHub, Bitbucket, GitLab, SourceForge, RhodeCode, Azure DevOps Server, or **Deployments (any CI)**. Commits and pull requests are tied to the work item their message cites by its reference: `NW-14 fix the queue` lands on NW-14's changelog. Each part of the organisation can number its work in its own prefix - Support's items SUP-1, SUP-2, Sales' SAL-1 - set by an admin under Organising and inherited by everything beneath it; the top of the tree uses the organisation's own, taken from its name when it was created (the demo's is NW). Changing a prefix affects only items raised afterwards, and a commit citing an older reference still lands on its item.

1. **Generate secret**. It is shown once; copy it.
2. In the service, add a webhook to the **Webhook URL** shown (prefixed with DilliC's own address), with the secret. The screen says how each service sends it: GitHub and GitLab take it as the webhook's secret; Bitbucket and Azure DevOps send it in a header; CI signs the body with it.
3. Optionally, give the **Repository** (`owner/name`), the **Server** for a self-hosted one, and a **Token** that can write, and switch on what DilliC may do back: comment on a pull request or commit it ties to an item; create a branch for an item (from the Terminal tab); take an item's next step when its pull request merges. Services that cannot do one say so.
4. **Save repository settings**.

Rotate secret issues a new one; **Disable** stops deliveries being accepted. A CI pipeline reports a deployment by posting the environment and status (and optionally an id, the commit, a URL, a description citing items, and who ran it), signed with the secret.

Environments

Environments have a tab of their own in Admin, after **Workflow & approvals**, because what is set on them is change-management policy. They are explained here because the pipelines of the section above are what fill the list. Environments are the places work runs, in order along the path to production. The order is what answers "how far has this got". **Add one** with a name and the **key** the pipeline calls it by; mark which is **Production**, and set its **Authorisation** to one of three positions. *Not asked* flags nothing. *Recorded* flags an arrival no authorised change request covers. *Gated* also holds a pipeline that asks the deployment check first; one that does not ask is let through and its arrival is recorded as a bypass, because DilliC is not in the path. An environment the pipeline deploys to that is not listed is added at the bottom automatically. One with deployments recorded against it is renamed rather than removed.

Outbound webhooks

DilliC can post to another system when things happen here:

1. **Add**. Say **What it is** ("Slack #delivery") and **Where to post it** (an https address; a plain http one is refused, because a delivery says what happened to whose work).
2. Choose the **Format**: **Signed JSON** for a receiver of your own, or **Slack** or **Microsoft Teams** to post straight into a channel.

3. Tick the **Events**. Families such as `work_item.*` cover everything of a kind.
4. **Add subscription**, and copy the signing secret now; it is not shown again.

Each delivery is signed: `X-DilliC-Signature` carries `sha256=` and an HMAC of the timestamp and body, with the timestamp and event in their own headers. A developer writing a receiver uses these to check that a delivery is DilliC's; a post into Slack or Teams needs none of it. Deliveries are sent within a minute and retried up to five times. **Deliveries** shows each attempt and its outcome; **Turn off** pauses a subscription.

Straight into a channel. A subscription in the Slack or Teams format needs no receiver: each event arrives as one sentence, the person's name and what the audit trail says they did ("Priya Nair: Raised CHG-0012: Rotate the signing key"), with the event's name and time beneath and a button that opens the item. For Slack, add the **Incoming Webhooks** app to the channel and paste the URL it gives you. For Teams, open **Workflows**, choose **Post to a channel when a webhook request is received**, and paste the URL it gives you; the delivery is an Adaptive Card, which that workflow posts as it stands. The signature headers travel all the same, and the chat tool ignores them.

Apps

An app acts in DilliC as a person of its own, audited under its own name and reaching only what that person's roles reach. **Install an app** with its name, vendor, https address, and optionally the scopes it may use; copy its API token and signing secret when shown. Give its person a role under People. An app can add **panels** to work items: rows it pushes, or its own page in a sandboxed frame. **Switch off** pauses an app; **Uninstall** deactivates its person.

Two-way tracker sync

Sync keeps one DilliC board in step with one project in another tracker. DilliC syncs with fifty-three: Jira, GitHub Issues, GitLab Issues, Azure Boards, Rally, Linear, Asana, Trello, Microsoft Planner, ServiceNow, Zendesk, BMC Helix ITSM, Smartsheet, Notion, and thirty-nine more.

1. **Connect a tracker**. Name the connection, and choose the tracker and the board.
2. Give the tracker's address, the project (key, id, or name, as the form asks), an account email where needed, and an API token. Trello's project is its API key and the board's id, with a slash between. Planner's is the tenant, the application's id and the plan's id, and its token is the application's client secret; the application needs the Tasks.ReadWrite.All permission.
3. Choose **Which way**: **Push only** (DilliC writes out and never reads back; the safe one to start on), **Pull only**, or **Both ways**.
4. **Connect**, then map statuses under **Statuses**: each line reads "when it is that over there, it is this here".
5. **Sync now**, or wait: syncs run every five minutes.

Titles, descriptions, and the link always travel; statuses travel once mapped. Forty-one trackers take a status as well as give one, and for those you mark which of the tracker's statuses DilliC writes back. Miro, Teamwork, Nifty, Zoho Sprints and IBM Targetprocess give a status and take none. GitHub and GitLab have two statuses, open and closed; Planner has three, Not started, In progress and Completed. Where both sides changed an

item, the edit made in DilliC stands, and the conflict is logged. The next sync writes DilliC's edit over the other tracker's, which is never pulled back; to keep the other edit, copy it into the item before then. A pulled move into a gated status is held and logged. A pull updates only items already linked, unless the connection is told to bring in what is new. An item imported from the same tracker is linked to the issue it came from on the first sync, so no second issue is made for it. **What the syncs have been doing** lists every push, pull, conflict, hold and failure. **Disconnect** removes the links and mappings and changes nothing on either side.

A connection does more when you ask it to.

- **Bring in what is new at the far end**, on the form, makes a work item of each issue a pull finds that no item is linked to. **Bring everything in** does it once for the whole project: it shows how many items it would make, and the first twenty, and makes nothing until you confirm.
- **Carry comments**, on the form, sends each comment on a linked item to its issue and brings each comment on the issue to the item, once. Comments travel from the moment you tick the box; what was said before stays put. Fourteen trackers carry comments, and the form offers the box only for those.
- **Links** lists the connection's links, removes one, and adds one from an item's reference and the issue's key. Link work that both tools already track before the first sync, so that the sync makes no duplicates.
- **Address to call** makes a webhook address and shows its secret once. Give both to the tracker, and DilliC syncs within seconds of a change there and no longer waits for the clock.
- **Pause this connection**, on the form, stops its syncs and keeps its links and its mapping.

Two trackers have limits of their own. Planner records when a task was made and when it was completed, and no other change, so a task renamed or started in Planner is not pulled. A Zendesk ticket's description is its first comment and is written once, so a later edit to the description in DilliC does not reach the ticket.

A service desk's form names its own things. BMC Helix asks for a user name and a password, and for the company and the person new records are raised for. SysAid and HaloITSM ask for an application's client id and secret. TOPdesk asks for an operator's login and an application password. The form labels each field for the desk chosen.

Seven more desks sync: Freshdesk, Zoho Desk, HubSpot Service Hub, Dynamics 365 Customer Service, GLPI, ConnectWise PSA and Autotask PSA. Each keeps work items, requests or incidents in step with its tickets. Most of them make no ticket for nobody, so the connection names who a ticket is for. Freshdesk asks for a requester's address. Zoho Desk asks for the organisation and department ids, with a slash between. Dynamics asks for the tenant, the application's id and the account's id. ConnectWise asks for the board's id and the company's, and Autotask for the company's id. HubSpot and GLPI need nothing named: HubSpot takes the first ticket pipeline, and GLPI the entity the user works in.

Change requests sync too. For ServiceNow, Jira Service Management, Freshservice, BMC Helix ITSM, ManageEngine ServiceDesk Plus, Ivanti Neurons for ITSM, SolarWinds Service Desk, SysAid, TOPdesk, HaloITSM, EasyVista, OpenText SMAX and Salesforce Service Cloud, the form offers **What it keeps**. **Work items** joins a board to the desk's tickets. **Change requests** joins a part of the organisation's change requests to the desk's change records. Choose when you connect; the choice cannot be changed after, so make a second connection for the other. The title, the description, the planned window and the mapped status travel, and the risk is written out where the desk has a field for it. A pulled status moves a change only where

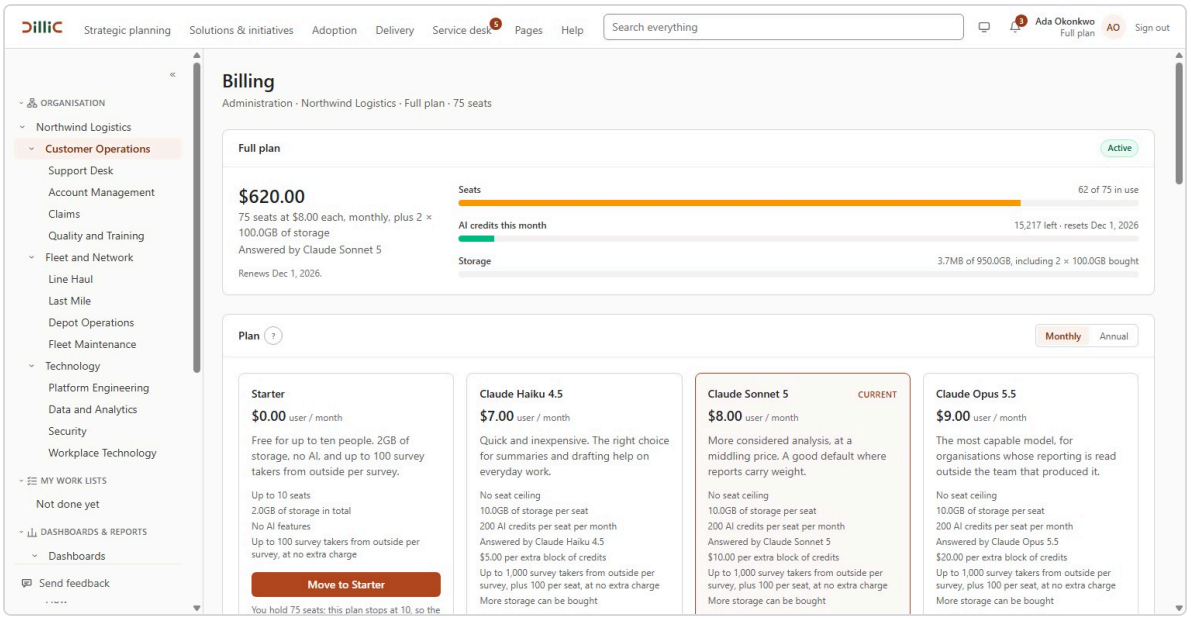
a person could make that move unaided. A move that needs an approval, or plans the change has not, is held and logged with the reason; make the move in DilliC, and the next sync finds the two sides agreeing.

Initiatives sync with Aha! and Jira Align. For those two the form also offers **Initiatives** under **What it keeps**, which joins a part of the organisation's initiatives to the initiatives of a product in Aha!, or to the epics of a program in Jira Align. The name, the description and the mapped status travel; the budget, the measure and the business case stay in DilliC. An initiative is sent once it is approved, so a proposal never appears on a roadmap. Map the far end's statuses to an initiative's five. A status from the far end never approves an initiative and never withdraws an approval: that move is held and logged, and somebody makes it in DilliC. The portfolio kanban is not moved by a sync.

Requests and incidents sync with other service desks. For twenty-one desks the form offers **Incidents** under **What it keeps**, and for sixteen of them **Requests** as well. ServiceNow, Freshservice, BMC Helix, Ivanti and SMAX keep a request as an order from a catalogue, so no request is sent to them. A connection carries one kind; make a second for the other. The subject and the mapped status travel both ways. What the person asked for goes out as the ticket's description and is not taken back. A request is first sent while it is new, open or waiting, and one already linked is kept in step until it closes. Map the far end's statuses to a request's five. A status from the far end is applied, and an internal note on the request says what changed. Tick **Bring in what is new at the far end** and each unlinked ticket becomes a request or an incident on the first queue; open a queue first. On ServiceNow, Jira Service Management, Freshservice, Zendesk, SMAX, Salesforce, Freshdesk, Zoho Desk, GLPI and ConnectWise, **Carry comments** sends each public reply to the ticket and brings each comment back as an internal note, so nothing is mailed to the requester.

CHAPTER 22

Billing



Billing: the plan and its meters, and the plans to move between.

Plans and prices

The same list is public at **/pricing**, linked from the sign-in page, for anybody comparing products before they have an account.

PLAN	PER PERSON	INCLUDES
Starter	Free	Up to ten people, with any email address; 2GB of storage; no AI; surveys sent to up to 100 people outside the organisation each.
Full, Claude Haiku 4.5	$7 a month, $70 a year	No seat limit; 10GB of storage per seat; 200 AI credits per seat each month; surveys sent to up to 1,000 people outside the organisation each, plus 100 per seat. The tiers differ in the AI model that answers and the price of extra credits.
Full, Claude Sonnet 5	$8 a month, $80 a year	
Full, Claude Opus 5.5	$9 a month, $90 a year	

These are the launch prices, and an organisation that subscribes to the full plan at them keeps them. DilliC ends launch pricing when it chooses; an organisation that subscribes later pays $12, $13 or $14 a month, or $120, $130 or $140 a year. Before launch DilliC is a beta: every plan is free, for any number of seats, and comes with no promises.

The **plan** card shows the recurring total, when it renews, and meters for seats, AI credits this month, and storage. Under **Plan**, choose monthly or annual and **Move to** another plan. A move starts a new term that day, and whatever was left of the term already paid comes back as a credit on the new invoice. Moving to the starter plan with more than ten people brings the organisation down to ten. **Cancel at the end of the term** drops to the starter plan when the term ends; until then everything stays available, and **Keep the subscription** undoes it.

Seats

A seat is one active person. Set the number and **Save**. Added seats are charged for the rest of the term and usable at once; removed seats come off at renewal. Seats cannot go below the number of people active: deactivate someone under People first.

AI credits

AI analysis and AI-assisted writing use credits. A credit is a thousand tokens of what the model reads, with what it writes counting five times over. The full plan includes 200 per seat each month, used first; they reset monthly and do not carry over. Buy more in **blocks of 500** ($5, $10, or $20 a block by tier), invoiced separately and lasting a year. The card shows what is left and what each feature has used; **Usage** lists every AI call, who made it, and what it cost.

Storage

Attachments and files count against storage. The full plan can add blocks of 100GB at $10 a month or $100 a year each, billed with the term. The page warns if the organisation would be over its allowance at renewal.

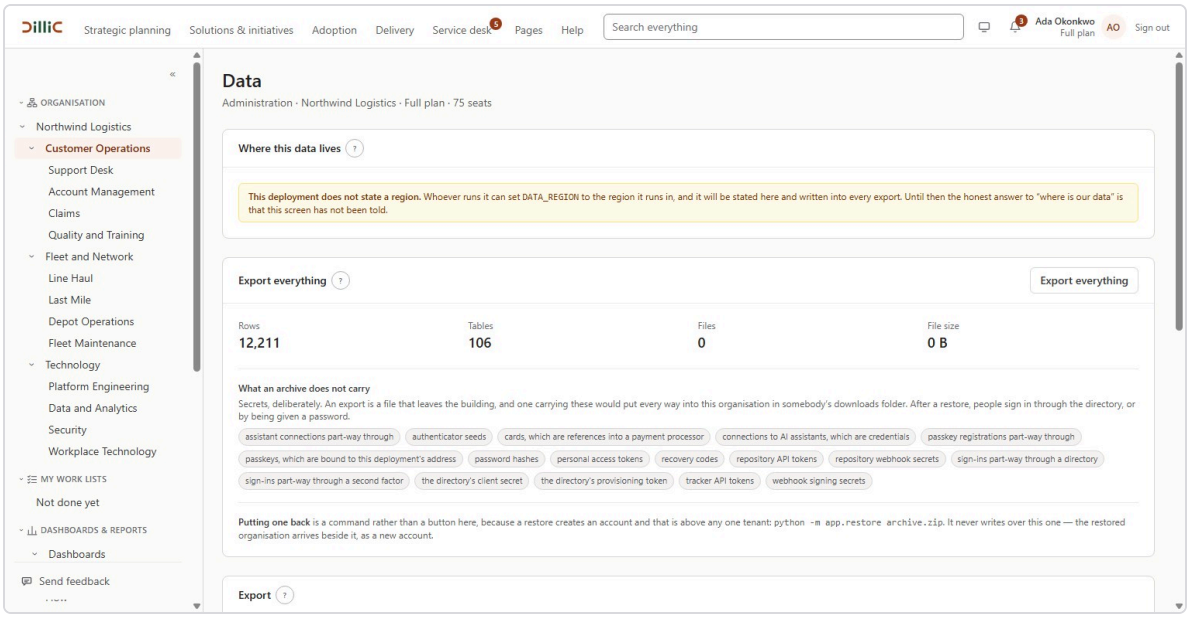
Payment and invoices

Add a card opens the payment processor's own page; DilliC keeps only the card's brand, last four digits, and expiry. **Use for renewals** makes a card the one charged. Invoices are issued in advance, due in 14 days, and never rewritten: corrections appear on the next invoice. Each can be charged to the card, paid on the processor's page, or marked paid by hand. A card declined four times stops being retried, and the account is marked past due; nothing is switched off.

CHAPTER 23

Data and feedback

Data and residency



Data: where the data lives, and exporting all of it.

Where this data lives states the region the service runs in, where every row and file of the organisation is held.

Export everything downloads one zip of every row and file the organisation holds, with a manifest of when, where, and which version. It shows how many rows, tables, and files there are. Deliberately, it leaves out secrets: passwords, authenticator seeds and recovery codes, access and provisioning tokens, directory and tracker secrets, webhook secrets, and cards.

The DilliC team can restore an export; write to support@dillic.app and say which one. A restore never overwrites: the restored organisation arrives beside the original as a new one, with its connections switched off and unsent mail retired.

> **ERASING A PERSON**
>
> A person outside the organisation can be anonymised from any survey's invitations page (chapter 6), or can delete their own details from their link. A member who leaves is deactivated under People, and can be anonymised there too (chapter 14).

Export and import

One board at a time, on the same Data tab as the whole. **Export** writes everything on one entity's board to a file: in DilliC's own format (the one to use for a backup or a move), or in the columns of any of twenty-seven trackers, Jira and ServiceNow among them, with statuses translated by meaning, as CSV or JSON. No file carries comments, attachments, links, or history.

Import reads a tracker's export as work items:

1. Choose the **Format**, the entity, and optionally the initiative it lands under.
2. Upload the CSV, or paste the rows. The first line is the header; the hint lists the columns read.
3. **Dry run**. DilliC says how many items would be created, previews them, and warns about anything it could not match (people, priorities, and types are matched to what the organisation has; unmatched ones are left off).
4. **Import**, which is offered only after a dry run passes.

Feedback

This page lists every bug report and suggestion people in the organisation have sent with **Send feedback**: what it says, who sent it, from which screen and platform, its attachments, and whether it has reached the DilliC team's service desk ("Sent as …") or is waiting.

APPENDIX A

The query language

One language asks questions of work everywhere: the board's Ask, the backlog, lists, quick filters, reports, and automation.

A question is one or more **conditions** joined by `and` and `or`. `not` negates, brackets group, and `and` binds tighter than `or`. A condition is a field, a test, and usually a value. Fields and values are written the way the screen shows them, ignoring capitals. Put a value in quotes if it has spaces or brackets. A list in brackets, `(a, b)`, means any of them.

KIND OF FIELD	TESTS
Text	`contains`, `does not contain`
Choice or person	`is`, `is not`, `is empty`, `is not empty`
Number, money, duration	`at least`, `at most`, `is empty`, `is not empty`
Date	`before`, `after`, `in the last` N days, `not in the last` N days, `is empty`, `is not empty`

FIELD	NOTES
title	Text.
type	The rung: Story, Task, Defect…
status	By name, across workflows.
state	One of the seven states.
priority	
points, size	The estimate.
original_estimate, remaining_estimate, time_spent	Durations, written `1d 2h`.
version	
assignee, reporter	`me` means whoever is reading.
epic, initiative	
component, label	`is` means carries any of them; `is not` carries none.
sprint	Also `current`, `next`, `after next`.
planning_interval	Also `this period`, `next period`, `after next period`.
score	RICE, ICE, or WSJF score.
due, created, updated	Dates.
cf:*name*	A custom field, such as `cf:regulator`.

Examples

- `priority is Critical and assignee is me`
- `status is not Done and (component is Billing or label is tech-debt)`
- `updated not in the last 14 and points is empty`
- `sprint is current and state is "Waiting on something"`
- `cf:regulator is FCA and type is Defect`

When a question cannot be read, DilliC says what is wrong: a field it does not know, a test a field does not take, or two conditions with no `and` or `or` between them.

APPENDIX B

States, statuses and the default workflows

The seven states

STATE	MEANING
Backlog	Waiting to be looked at.
Ready	Ready for somebody to pick up.
In progress	The work is in progress.
Waiting on something	Blocked, or with somebody outside the team. Not counted as flow.
Being reviewed	Somebody is judging it rather than advancing it.
Complete	Finished. A burndown stops counting it here.
Cancelled	Dropped. A burndown stops counting it, and it is kept out of velocity.

The default delivery workflow

STATUS	STATE	MOVES OUT
Backlog	Backlog	Refine → Ready
Ready	Ready	Start → In progress; Send back → Backlog
In progress	In progress	Ready for review → In review; Block → Blocked; Put back → Ready
Blocked	Waiting on something	Unblock → In progress
In review	Being reviewed	Sign off → Done (gated: definition-of-done sign-off by one person, not the author); Changes requested → In progress
Done	Complete	Reopen → In review
Cancelled	Cancelled	Reopen → Backlog. Cancel is allowed from anywhere.

Default work-in-progress limits: In progress 3, In review 2. Your organisation may have changed any of this.

The default change request workflow

New → Assessment → Authorisation → Scheduled → Implementation → Post-implementation review → Closed. Authorisation to Scheduled is gated by the Change Advisory Board (two leaders, not the requester).

A standard change may go straight from New to Scheduled. Cancel is allowed from anywhere and reopens to New; Closed is final.

APPENDIX C

Glossary

TERM	MEANING
Entity	A part of the organisation: a company, division, team, or any level you defined.
Level	A rung of the hierarchy: Company, Division, Team.
Weight	How much an entity counts among its siblings in a composite.
Role	A named set of rights, held on an entity and everything beneath it.
Module	A part of DilliC the organisation switches on or off.
Factor	Something that matters to an entity, worded neutrally so it can be scored.
Objective	What you intend for a factor, with a KPI, target, and date.
Measure/KPI	What gets counted, from the organisation's library. Renameable under Vocabulary.
Category / goal	A grouping of factors, and the aim they serve.
Applicability	How far people agree a proposed factor applies, 0–100%.
Assessment	A survey of the accepted factors and its results.
Current state	An assessment of how things are today.
Desired future state	An assessment of how things would be if a solution were delivered.
Composite	Child entities' assessments combined, by weight, at the parent.
Combined score (∗)	All of an assessment's answers taken together.
Agreement	How far respondents agreed about a factor; shown as colour and in words.
Theme	A grouping of survey comments, made by hand.
Assertion	A claim about a factor that root cause analysis starts from.
Cause / root cause	An answer to "why?"; a root cause has nothing beneath it.
Initiative	A piece of change with a budget, a measure of whether it is effective and targets for whether it is adopted, broken down into work.
Adoption target	A measure/KPI, a value, and a date that say whether the people an initiative happens to have changed.
Solution	One or more initiatives put forward as an answer, assessable as a desired future state.

Spend	Money recorded against a budget: actual, commitment, or forecast.
Work item	An epic, story, task, defect, or sub-task.
Rung	A kind of work item and its place in the breakdown.
Status / state	A column's word on a board, and which of the seven states it means.
Workflow	The statuses work moves through and the moves allowed between them.
Approval gate	A move that waits for enough of the right people to approve.
WIP limit	The most work a board column may hold.
Sprint	A fixed period of planned work.
Version	A release that work is aimed at.
Change request	A request to put a change into production, with its plans, approval, and outcome.
Environment	A place work is deployed to, in order towards production.
Queue	Where service desk requests arrive, with its promised times.
List	A saved question about work, asked afresh each time.
Gadget	One question on a dashboard.
Value stream	A budget that funds epics on the portfolio.
Planning interval	A train's shared period of iterations, planned together (SAFe).
AI credit	The unit AI features are paid in.

Words from agile delivery and planning

DilliC borrows these from the methods teams already use. They mean here what they mean there.

TERM	MEANING
Kanban	Work flowing continuously across a board, with a limit on how much each column may hold.
Scrum	Work planned into sprints of a fixed length.
Scrumban	Kanban's flow and limits, planned in sprints.
Epic	A large piece of work under an initiative, made of stories.
Story	A piece of work small enough to finish in a sprint, written from the point of view of whoever it is for.
Acceptance criteria	What must be true for a story to count as done.
Backlog	The work not yet started, in the order the team means to take it.
Points	A team's own unit for how big a piece of work is, compared with its other work. Not hours.
T-shirt size	A rougher estimate for epics: small, medium, large.
Planning poker	Estimating together: everyone picks a number in secret, and all are shown at once.
Capacity	How many points a team, or a person, can take on in a sprint.
Velocity	How many points a team finishes in a sprint, on average.
Burndown, burnup	Charts of a sprint: the points left, falling towards zero, or the points done, rising towards the total.
Cycle time	How long a piece of work takes from start to finish.
Flow efficiency	The share of the cycle time spent being worked on rather than waiting.
Cumulative flow, control chart	Charts of how much work sits in each state over time, and of how cycle time varies.
Stand-up	A team's short daily meeting about what is moving and what is blocked.
Retrospective	A team's look back at a sprint: what to keep and what to change.
SAFe	The Scaled Agile Framework: a way for many teams to plan together. Chapters 10 and 19.
Train	In SAFe, a group of teams that plans and delivers together.
RICE, ICE, WSJF	Three ways of scoring work to put it in order. Chapter 16 says what each multiplies and divides.
SMART	Of an objective: specific, measurable, attainable, relevant and timebound.

Strength, weakness, opportunity, threat	The four words of a SWOT analysis, which an assessment's matrix is drawn from.
Incident, problem	On the service desk: something that went wrong, and the cause behind several such.
ITIL	A widely used body of practice for running IT services, from which change requests and the service desk take their words.

Words from IT

These appear mostly in part two. Chapters 20 and 21 open with the same words in context.

TERM	MEANING
Directory	The system that holds an organisation's work accounts: Okta, Microsoft Entra ID, Google Workspace.
Single sign-on	Signing in to DilliC with the work account, through the directory, with no second password.
SAML 2.0, OpenID Connect	The two standard ways a directory vouches for a person signing in.
SCIM	The standard by which a directory tells DilliC who has joined and who has left.
Provisioning	Creating and deactivating accounts automatically, from the directory.
Second factor	Something besides a password that proves who you are, such as a code from an app on your phone.
Authenticator app	A phone app that shows a fresh six-digit code every half minute.
Recovery code	A code kept for the day the phone is lost. Each works once.
Domain	The part of an email address after the @, such as acme.com.
DNS, TXT record	DNS is the internet's register of domains. A TXT record is a line of text published there, which only whoever controls the domain can add.
API	The way a program, rather than a person, reads and changes what DilliC holds.
Access token, API token	A password made for a program. Shown once, and revocable.
Scope	A limit on what a token or an assistant may do.
Webhook	A message one system sends another over the web the moment something happens.
Secret, signature	A secret is a password two systems share. A signature is the proof, worked out from the secret, that a message is genuine.
Repository, commit, pull request	Where developers keep code, one saved change to it, and a change offered for review.
CI, pipeline	Continuous integration: the service that builds, tests and deploys code by itself, and the steps it runs.
Deployment	Putting a version of the code to run in an environment.
Connector, MCP	How an AI assistant is joined to DilliC. MCP is the standard assistants use for it.
CSV	A plain file of rows and columns that any spreadsheet can save.

APPENDIX D

The service desk by email

The portal is one door to the service desk, and email is the other. DilliC reads the mail itself, so there is no mail server to set up: every public queue has an address, and every letter the desk sends can be answered by replying.

> **WHO DOES THIS**
> Anyone who may configure the desk can hand out a queue's address. Using an address on your own domain takes one forwarding rule, made by whoever runs your organisation's mail.

The address to hand out

Open **Service desk** and find the queue's card. Under the portal address, *By email* shows the address DilliC made for the queue. A letter sent to it becomes a request on that queue, as the portal would make. Like the portal address, it works only for a public queue.

Put that address wherever people look for help: a signature, an intranet page, the foot of an invoice. Nothing else needs doing.

An address of your own

An organisation that would rather hand out `help@acme.com` can do so with one forwarding rule.

1. Edit the queue and enter `help@acme.com` as its **Email address**.
2. The card now shows two lines: *By email*, with your address, and *Forward it to*, with the address DilliC made.
3. Ask whoever runs your mail to forward `help@acme.com` to the address under *Forward it to*.

What becomes of a letter

A LETTER	BECOMES
To a queue's address	A new request on that queue, with its attachments. The sender receives an acknowledgement with a portal link.
In reply to a letter from the desk	A reply on that request's thread, marked *by email*. Each letter the desk sends carries a reply address that belongs to one request, so answering is pressing Reply.
With a request's reference in its subject, such as "SD-12"	A reply on that request, if the sender is the person who raised it.
In reply to a closed request	A new request on the same queue, carrying the letter.
Anything else	Set aside. DilliC does not guess, because a guess could put one person's letter on another's request.

The same letter arriving twice is filed once. A private queue takes no letters, and an organisation with the service desk module switched off answers neither its addresses nor its portal.

Checking it

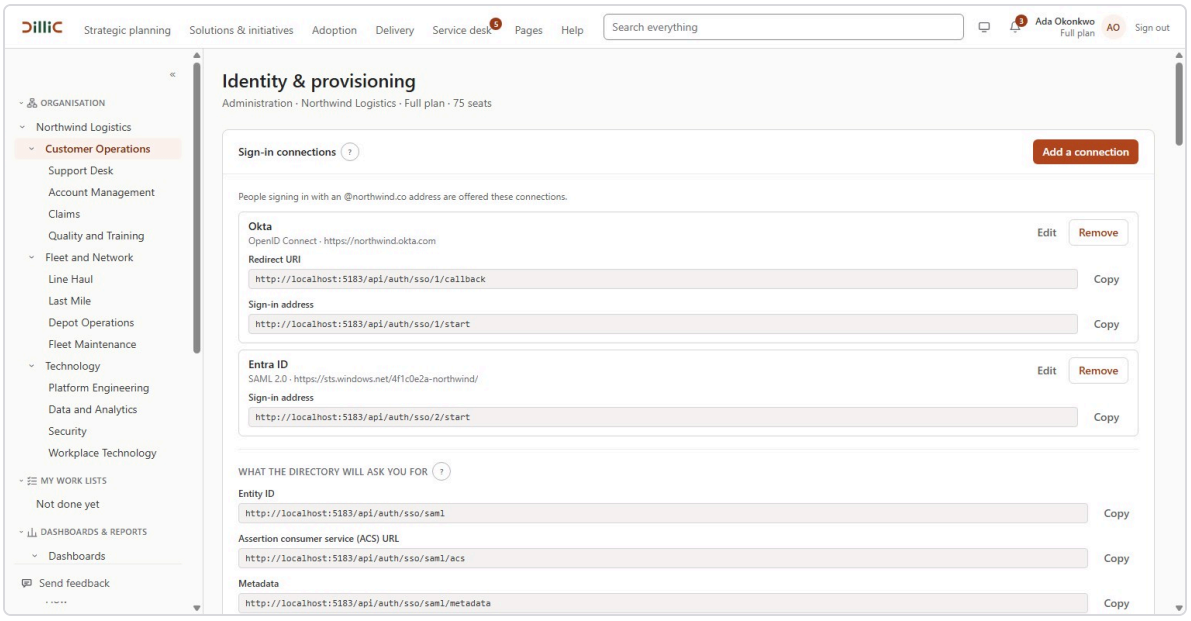
1. Send a letter to the queue's address from an address outside the organisation.
2. Within a minute or two it should appear on the desk as a request marked *by email*, and the acknowledgement should arrive.
3. Reply to the acknowledgement. The reply should appear on the request's thread.

If you forward an address of your own and the first step fails, send a letter straight to the address DilliC made. If that one arrives, the forwarding rule is at fault. If a card shows no address under *By email*, tell the DilliC team with **Send feedback**, and use the portal address meanwhile.

APPENDIX E

Connecting Okta, Entra ID and Google Workspace

Chapter 20 says what each setting on Identity & provisioning means. This appendix says where each value comes from in the three directories most organisations run, and where it goes. Keep two windows open: DilliC's Admin → Identity & provisioning, and the directory's own admin console. Prove the domain first; until then the sign-in page offers the connection to nobody.



Two saved connections, each with the addresses its directory asks for, above the details every SAML directory needs.

Choosing a protocol

DIRECTORY	SIGN-IN	PROVISIONING (SCIM)	USE
Okta	SAML 2.0 or OpenID Connect	Yes	SAML, with SCIM on the same app
Microsoft Entra ID	SAML 2.0 or OpenID Connect	Yes	SAML, with SCIM on the same enterprise application
Google Workspace	SAML 2.0 or OpenID Connect	No	OpenID Connect

SAML suits Okta and Entra ID because one application then holds both sign-in and provisioning, and it carries a job title that their OpenID Connect tokens leave out. Google provisions only the applications in its own catalogue, so a Google organisation adds people in DilliC or lets the connection create them on first sign-in. Google's SAML sends no full name in one attribute, which makes OpenID Connect the better door there.

What DilliC gives the directory

VALUE	WHERE DILLIC SHOWS IT	ASKED FOR BY
Entity ID	**What the directory will ask you for**	Every SAML directory, as the audience or identifier
Assertion consumer service (ACS) URL	**What the directory will ask you for**	Every SAML directory, as the single sign-on or reply URL
Redirect URI	Under each saved OpenID Connect connection	Every OpenID Connect directory
Sign-in address	Under each saved connection	A directory's app tile, so it starts the sign-in at DilliC
Base URL and token	**Provisioning (SCIM 2.0)**	Okta and Entra ID provisioning

Copy each value with its **Copy** button; a directory compares them character for character. The redirect URI and the sign-in address name the connection, so they appear only once it is saved. For OpenID Connect, therefore, save the connection first with **Offered on the sign-in page** unticked, the issuer, and any client ID; copy its redirect URI into the directory; then edit the connection to enter the real client ID and secret and tick it on.

DilliC answers only sign-ins it started. A directory that posts an assertion when someone clicks its tile is refused, so point the tile at the connection's sign-in address, or hide it. Every directory below signs the assertion, the response, or both; DilliC accepts any of these.

The steps below name each attribute `email`, `name` and `title`, which are the names DilliC reads when **Email claim**, **Name claim** and **Title claim** are left as they are. A directory's menus move from year to year; the names of the fields change less.

Okta

Sign-in over SAML

1. In the Okta Admin Console, open **Applications** → **Applications**, choose **Create App Integration**, pick **SAML 2.0**, and name the app DilliC.
2. Under **App visibility**, tick **Do not display application icon to users**. Okta's tile would post an assertion DilliC did not ask for.

3. On **Configure SAML**, paste DilliC's ACS URL into **Single sign-on URL** and leave **Use this for Recipient URL and Destination URL** ticked. Paste the Entity ID into **Audience URI (SP Entity ID)**. Set **Name ID format** to EmailAddress and **Application username** to Email.
4. Under **Attribute Statements**, add `email` with the value `user.email`, `name` with `user.firstName + " " + user.lastName`, and `title` with `user.title`. Finish.
5. On the app's **Sign On** tab, open **View SAML setup instructions**. It lists the **Identity Provider Single Sign-On URL**, the **Identity Provider Issuer**, and the **X.509 Certificate**.
6. In DilliC, **Add a connection**, protocol **SAML 2.0**, named Okta. Paste the issuer into **Identity provider entity ID**, the single sign-on URL into **Sign-on URL**, and the certificate into **Signing certificates**. **Save the connection**.
7. On Okta's **Assignments** tab, assign the people or groups who use DilliC. Okta refuses a sign-in from anyone else.

Sign-in over OpenID Connect

1. In DilliC, save a connection named Okta, protocol **OpenID Connect**, switched off, with the **Issuer** `https://your-org.okta.com` and any client ID. Copy its redirect URI and sign-in address.
2. In Okta, **Create App Integration**, pick **OIDC - OpenID Connect** and **Web Application**. Leave **Authorization Code** as the grant type.
3. Paste the redirect URI into **Sign-in redirect URIs**. To show a tile, paste the sign-in address into **Initiate login URI** and set **Login initiated by** to Either Okta or App.
4. Assign people or groups and save. Okta shows the **Client ID** and the **Client secret** on the app's **General** tab. DilliC sends PKCE on every sign-in, so **Require PKCE** may be ticked.
5. In DilliC, edit the connection: enter the client ID and the secret, tick **Offered on the sign-in page**, and save.

Okta's token carries no job title. Provisioning over SCIM supplies one, or choose SAML.

Provisioning over SCIM

1. In DilliC, under **Issue a token**, name it "Okta provisioning", **Issue it**, and copy it. It is shown once.
2. In Okta, on the DilliC app's **General** tab, edit **App Settings**, set **Provisioning** to SCIM, and save. A **Provisioning** tab appears.
3. On **Provisioning** → **Integration**, choose **Edit**. Paste DilliC's Base URL into **SCIM connector base URL**. Set **Unique identifier field for users** to `userName`.
4. Under **Supported provisioning actions**, tick **Push New Users**, **Push Profile Updates**, and **Push Groups**. Set **Authentication Mode** to HTTP Header and paste the token as the bearer token.
5. **Test Connector Configuration**, then save.
6. On **Provisioning** → **To App**, choose **Edit** and enable **Create Users**, **Update User Attributes**, and **Deactivate Users**. Save.

7. To send groups as roles, set **Where they hold it** on the Okta connection in DilliC, then add the groups on Okta's **Push Groups** tab.

Okta now creates each assigned person in DilliC, keeps their name, email and title in step, and deactivates them when they are unassigned or leave. A pushed group arrives as a role that grants nothing until an administrator gives it rights under Roles.

Microsoft Entra ID

Sign-in over SAML

1. In the Microsoft Entra admin center, open **Enterprise applications**, choose **New application**, then **Create your own application**. Name it DilliC and pick **Integrate any other application you don't find in the gallery**.
2. Open **Single sign-on** and pick **SAML**. Under **Basic SAML Configuration**, choose **Edit**; paste DilliC's Entity ID into **Identifier (Entity ID)** and the ACS URL into **Reply URL (Assertion Consumer Service URL)**. Save.
3. Under **Attributes & Claims**, choose **Edit**, then **Add new claim** three times, each with the namespace left blank: `email` from `user.mail`, `name` from `user.displayname`, and `title` from `user.jobtitle`.
4. Under **SAML Certificates**, download **Certificate (Base64)**. Under **Set up DilliC**, copy the **Login URL** and the **Microsoft Entra Identifier**.
5. In DilliC, **Add a connection**, protocol **SAML 2.0**, named Entra ID. Paste the identifier into **Identity provider entity ID**, the login URL into **Sign-on URL**, and the certificate file's text into **Signing certificates**. **Save the connection** and copy its sign-in address.
6. Back in **Basic SAML Configuration**, paste the sign-in address into **Sign on URL**. The DilliC tile in My Apps then starts the sign-in at DilliC.
7. Under **Users and groups**, assign the people or groups who use DilliC.

Entra ID renews its signing certificate before it expires. Add the new certificate to the connection, under the old one with a blank line between, before making it active in Entra ID; remove the old one afterwards.

Sign-in over OpenID Connect

1. In DilliC, save a connection named Entra ID, protocol **OpenID Connect**, switched off, with the **Issuer** `https://login.microsoftonline.com/tenant ID/v2.0` and any client ID. Copy its redirect URI.
2. In Entra ID, open **App registrations** and choose **New registration**. Name it DilliC, choose **Accounts in this organizational directory only**, and under **Redirect URI** pick **Web** and paste the redirect URI. Register.
3. The **Overview** shows the **Application (client) ID** and the **Directory (tenant) ID**; the tenant ID belongs in the issuer.
4. Under **Certificates & secrets**, add a **New client secret** and copy its **Value**, not its Secret ID. Entra ID shows it once.

5. Under **Token configuration**, choose **Add optional claim**, token type **ID**, and tick **email**; accept the Microsoft Graph permission it offers. Without it the token carries no address.
6. In DilliC, edit the connection: enter the client ID and the secret value, tick **Offered on the sign-in page**, and save.

Provisioning over SCIM

1. In DilliC, issue a token named "Entra ID provisioning" and copy it.
2. In the DilliC enterprise application, open **Provisioning** and set **Provisioning Mode** to Automatic.
3. Under **Admin Credentials**, paste DilliC's Base URL into **Tenant URL** and the token into **Secret Token**. **Test Connection**, then save.
4. Under **Mappings**, open **Provision Microsoft Entra ID Users**. DilliC takes `userName` as the person's email address, and Entra ID fills it from `userPrincipalName`. Where the two differ, map `userName` from `mail` instead. Entra ID sends `jobTitle` as `title` already.
5. Under **Settings**, set **Scope** to Sync only assigned users and groups. Set **Provisioning Status** to On and save.
6. To send groups as roles, set **Where they hold it** on the Entra ID connection in DilliC, and assign the groups under **Users and groups**.

Entra ID provisions in cycles about forty minutes apart. **Provision on demand** sends one person at once, which is the quickest test.

Google Workspace

Sign-in over OpenID Connect

1. In DilliC, save a connection named Google, protocol **OpenID Connect**, switched off, with the **Issuer** `https://accounts.google.com` and any client ID. Copy its redirect URI.
2. In the Google Cloud console, choose or create a project inside the organisation. Open **Google Auth Platform** (formerly the OAuth consent screen), and under **Audience** set the user type to **Internal**.
3. Under **Clients**, choose **Create client**, application type **Web application**, named DilliC. Paste the redirect URI into **Authorized redirect URIs** and create it.
4. Copy the **Client ID** and the **Client secret**.
5. In DilliC, edit the connection: enter the client ID and the secret, tick **Offered on the sign-in page**, and save.

Internal matters. An external client lets anyone with a Google account finish the sign-in, and a connection that creates people on first sign-in would then create them. Internal limits it to the organisation's own accounts.

Without SCIM

Google sends no provisioning to an application outside its catalogue, so choose one of two ways to say who exists. Add people under People, by hand or by import, and leave **Create people on their first sign-in** off. Or tick it, with a role and a place, and let each person arrive the first time they sign in. Either way, deactivate a leaver in DilliC when Google suspends them: Google will not tell DilliC.

Checking it

1. In a private window, open DilliC's sign-in page and type an address at the organisation's domain. A button, **Sign in with** and the connection's name, appears above the password field.
2. Sign in through it. The directory asks for its own password and second factor, then returns you to DilliC, signed in.
3. Under **Two-factor policy**, the list shows the person as signing in through the directory.
4. For SCIM, assign a new person in the directory and send them. They appear under People with their title. Unassign them, and they are deactivated and signed out everywhere.

When DilliC refuses a sign-in, the page says why: an assertion issued for another service, a signature from a certificate the connection does not list, an issuer it does not name, an address nobody in the account holds. Each names the setting to fix. A redirect URI the directory does not know is reported by the directory, before DilliC is reached.

APPENDIX F

If something goes wrong

What people most often meet, the usual reason, and what to do. When DilliC refuses something it says why, and the refusal usually names the next step, so read it first.

Signing in

WHAT YOU SEE	THE USUAL REASON	WHAT TO DO
"This account has been deactivated"	An administrator switched the account off.	Ask an administrator to reactivate it under People. It needs a free seat (chapter 14).
"… is not set up in DilliC", after signing in through the directory	The directory knows you, and DilliC does not yet.	Ask an administrator to add you (chapter 14).
No Sign in with … button appears	Your email is at a domain the organisation has not proved, or the connection is not enabled.	Sign in with your password if you have one. Administrators: see Proving the domain (chapter 20).
The directory sign-in is refused	A setting on the connection does not match the directory.	The page says which. Appendix E, Checking it, lists them.
You have lost the phone with your authenticator app		Enter a recovery code in place of the six-digit code. If the codes are lost too, an administrator can reset your second factor (chapter 20).
Forgotten your password? sends nothing	You sign in through a directory, and your password lives there.	Reset it in the directory.
The link in your invitation no longer works	It is good for three days.	Ask for another with **Forgotten your password?** on the sign-in page.
The confirmation email for a new organisation has not come	It may be in junk mail. The organisation exists only once you follow the link.	Look there, then sign up again with the address spelt out carefully.

Something is missing

WHAT YOU SEE	THE USUAL REASON	WHAT TO DO
A tab or a button this guide shows is not there	Your role does not carry that right, or the module it belongs to is switched off.	Ask an administrator which (chapters 3 and 12).
There is no Administration group in the left pane	You are not an account administrator.	Ask one; chapter 12 says who they are.
There is no Overview tab	No assessment has been run on this part of the organisation.	It appears after the first one (chapter 4).
There is no Sprints tab	The board runs as Kanban.	Someone who may configure boards can change its methodology (chapter 9).
Search does not find a factor, a measure/KPI or a person	Search leaves those out.	Find a factor on Strategic planning, and a person wherever a person picker offers them.
Somebody is not offered when you type @	They cannot open the item, and a mention would tell them about it.	Ask whoever gives roles to give them one that reaches it.
The screen shows another team's work	Another entity is chosen in the Organisation tree.	Click your own. DilliC remembers it.
An epic is not on the board	Epics are containers and are never on a board or in a sprint.	Find it under Epics on the Backlog, or on the Timeline.

Surveys and assessments

WHAT YOU SEE	THE USUAL REASON	WHAT TO DO
Your answers vanished when you left the page	A survey saves no draft.	Answer it in one sitting.
Submit answers will not press	A factor is still unanswered.	Answer both questions for every factor, or tick N/A. The bar at the top counts them.
You cannot change an answer	Submitted answers are final.	Nothing, unless the factor is reset while the survey runs; then you are asked that one question again.
You were asked one question a second time	The factor was reset: it now asks something different, or its target moved.	Answer that question. The rest of your answers stand.
A new assessment says there is nothing to assess	This part of the organisation has no accepted factors.	Accept some first (chapter 5).
Somebody outside did not receive their invitation	The address was wrong, the letter bounced, or they had unsubscribed.	The invitations page says which. Use **Correct address** or **Send again** (chapter 6).
Two assessments cannot be compared factor by factor	A factor had a full reset between them, which starts a new baseline.	Compare from the later one onwards.
An assessment cannot be archived	It is open or paused.	Close or cancel it first.

Work and boards

WHAT YOU SEE	THE USUAL REASON	WHAT TO DO
A card will not drop into a column	The column is at its work-in-progress limit, or the workflow has no such move; columns you cannot reach are dimmed as you drag.	Finish something in that column first, or take the card through the steps the status menu offers.
A card you moved stays where it was, marked awaiting approval	The move is behind an approval gate.	Wait for the approvers. The item shows who may decide (chapter 9, Approvals on work).
The backlog shows no grip to drag rows by	The backlog is sorted, searched, asked, filtered or grouped.	Clear those, and the team's own order and the grips return.
A sprint will not start	Another sprint is running; only one runs at a time.	Complete the running one.
A board will not switch to Kanban	A sprint is running.	Complete it first.
Archive is not offered on an item	It is offered once the item is released or cancelled.	Finish or cancel the work first.
An initiative will not delete	Something still hangs off it.	The refusal names what. Remove that, or archive the initiative instead.
An initiative has no score	A part of the score is missing.	**Score it**, and fill in every part (chapter 16).
A commit does not show on its work item	The commit message does not cite the item's reference, such as NW-14; or the work has not been started.	Cite the reference in the message. Administrators: check the repository's webhook (chapter 21).

Notifications and email

WHAT YOU SEE	THE USUAL REASON	WHAT TO DO
You are not told about things	That kind of notification, or that place to receive it, is unticked; or you chose once a day.	Open your name, then Notifications (chapter 2).
You stopped hearing about an item	You stopped watching it, and DilliC does not start you again.	Press **Watch** at the top of the item.
The phone gets no notifications	"My phone" is unticked.	Tick it in your notification settings.
A chat address, token or secret cannot be seen again	Each is shown once and then kept unreadable.	Make a new one and replace the old.
A letter to a service desk queue raised no request	The queue is private, or a forwarding rule is at fault.	Appendix D, Checking it.

Connections, for administrators

WHAT YOU SEE	THE USUAL REASON	WHAT TO DO
Verify does not find the DNS record	A new record takes a few minutes to be seen, or it was added under the wrong name.	Compare the record with what Verify says it looked for, and try again shortly.
A webhook does not arrive	The address refuses it, or the subscription is turned off.	**Deliveries** shows each attempt and its outcome. A delivery is retried up to five times.
A status does not travel in a tracker sync	It is not mapped; or the move needs an approval here, and was held.	Map it under Statuses. *What the syncs have been doing* lists every hold and failure with its reason.
New issues in the other tracker do not appear	A pull updates only items already linked.	Tick **Bring in what is new** on the connection.
An edit made in the other tracker was overwritten	Both sides changed the item, and DilliC's edit stands.	The conflict is logged. Make the edit in DilliC.
The seat count grew unexpectedly	The directory is creating people on first sign-in.	Untick Creating people on first sign-in (chapter 20).
Somebody cannot be added or reactivated	Every seat is taken.	Add seats under Billing, or deactivate someone.

Still stuck

The (?) beside a heading explains what the thing is and why it behaves as it does. **Send feedback**, at the foot of the left pane, reaches the DilliC team from any screen, with the screen you were on attached. Or write to support@dillic.app.