Workflow Mapping — Setup Guide

Agile Analytics for Azure DevOps

> **Why this guide exists.** Workflow Mapping is the single most important setup step in Agile Analytics. It is the bridge between *your* Azure DevOps states and the analytics engine — every cycle-time, flow-efficiency, aging, WIP and forecast number is computed from it. Get this right once and every dashboard becomes accurate for your team. This guide explains the model, shows you exactly how to map common Azure DevOps processes (Scrum, Agile, CMMI, Basic and custom), and walks you through setup and verification.

Useful links

- 📖 Documentation home — https://ado-analytics.baytekdev.com/docs
- 🧭 Feature overview — https://ado-analytics.baytekdev.com/features
- 📊 Flow-metrics articles — https://ado-analytics.baytekdev.com/blog
- 🎬 Book a setup walkthrough / demo — https://ado-analytics.baytekdev.com/support/request

---

1. The big idea

Every team runs Azure DevOps a little differently. You might use out-of-the-box Scrum states (`New → Committed → Done`), the Agile template (`New → Active → Resolved → Closed`), or a rich custom board with states like `Ready To Test`, `Ready To Review`, `Ready To Accept`, `Ready To Release`. Agile Analytics does **not** force you onto a fixed process.

Instead, the tool defines **six canonical stages** that describe the universal shape of delivery work. You tell it which of *your* states belong in each stage. From that point on, the engine speaks your language: it knows when work "starts," when it's "waiting," and when it's "done," no matter what you call those states.

```
   NEW   →   QUEUED   →   IN PROGRESS   →   TEST   →   REVIEW   →   COMPLETE / DONE
(backlog)  (committed,    (active work)   (in QA)  (in review/   (delivered / closed)
            not started)                            acceptance)
```

You can map your states two ways — let the tool **auto-detect** them, or map them **by hand** — and any stage your team doesn't use can be marked **N/A**. Both are covered below.

---

2. The six stages, defined

Stage	What it represents	Typical Azure DevOps states
New	On the backlog, not yet committed to a sprint. *Excluded from all delivery metrics.*	`New`, `Approved`, `Proposed`, `To Do`
Queued	Committed to the sprint / ready to start, but **work has not begun**.	`Committed`, `Ready`, `Approved`
In Progress	Active development — someone is working on it. **This is where the cycle-time clock starts.**	`In Progress`, `Active`, `Doing`
Test	Development is done; the item is in QA / testing (e.g. PR merged, awaiting test).	`Ready To Test`, `Testing`, `In QA`
Review	Passed test; awaiting review or product acceptance.	`Ready To Review`, `Ready To Accept`, `Resolved`
Complete / Done	Delivered, accepted, or closed. **This is where the cycle-time clock stops.**	`Done`, `Ready To Release`, `Closed`, `Resolved`

Three of these stages — In Progress, Test and Review — are the "active work" stages. Together they define when an item is genuinely being worked. New and Queued are *pre-work*; Done is *post-work*.

---

3. How your states become metrics

This is the part worth understanding, because it determines what every number on your dashboards means.

Cycle Time — *first In Progress → first Done*

The cycle-time clock **starts the moment an item first enters any state you mapped to In Progress, Test or Review** (an "active" stage) and **stops when it first reaches a Done state.**

> ⚠️ **Important nuance.** Time an item spends in **New** or **Queued** is **excluded** from cycle time — it counts as *pre-work*. So if your team commits items to a sprint days before anyone starts them, that waiting time is **not** in your cycle time. Cycle time measures *active delivery time*, from first touch to done.
>
> If your team treats "committed" as "*we're working on it now*" (you have no separate in-progress state), then map your committed/working state to **In Progress** rather than Queued — see §5 (Scrum). That makes the clock start when work is committed, which is probably what you want.

Lead Time — *created → first Done*

Lead time is the full elapsed time from when the work item was **created** in Azure DevOps to when it reached **Done**. It includes all the waiting (backlog + queue), so it's always ≥ cycle time. Use it to answer "*how long from request to delivery?*"

Flow Efficiency — *active time ÷ total cycle time*

Flow efficiency = time in **active** stages (In Progress + Test + Review) ÷ total cycle time (active + waiting). A low number means work is sitting idle between stages. Items in **Queued** count as *waiting*; New and Done are excluded from the calculation.

WIP (Work in Progress)

WIP counts items **currently** in an active stage (In Progress / Test / Review). **Items still in Queued are *not* counted as WIP** — they're committed but not started.

Aging ("days here")

On the Aging board, each item shows how long it has sat in its **current** stage (lane-entry → today). This is how you spot work that's stuck.

Throughput, Commitment Reliability, Carryover, Scope Churn

These are **sprint-scope** metrics. The only workflow-mapping dependency is the **Done** stage: an item counts as *completed / throughput* when it reaches a state you mapped to Done. (Commitment, carryover and churn are derived from sprint membership at the Day-2 baseline, not from the workflow stages.)

What's excluded

- **New** stage states are excluded from cycle time, lead time, flow efficiency, WIP and throughput-completion. They're backlog — they shouldn't move your delivery numbers.
- **Cancelled work** (e.g. the Azure DevOps `Removed` state). Auto-detection treats `Removed` as *cancelled / out-of-flow* and leaves it unmapped, so it doesn't inflate throughput. If you *manually* place `Removed` under **Done**, it **will** be counted as completed — so only do that if you genuinely consider removed items "done."

---

4. Two ways to set it up

Option A — Let the tool detect it (recommended starting point)

When you first open Agile Analytics, it **auto-detects** a mapping by reading your project's real Azure DevOps states and their state categories. It recognises hundreds of common patterns across Scrum, Agile, CMMI and Basic — including **custom states** and **non-English** state names (e.g. *Listo, Concluído, Erledigt*). It maps the coarse shape automatically and surfaces anything it couldn't place so you can finish it.

Auto-detection is **safe**: it only ever seeds an *empty* mapping. It will **never overwrite** a mapping you've edited or built yourself.

Option B — Map your existing states by hand

In **Configuration** → **Workflow Mapping**, each of the six stages has:

- a **state dropdown** (pick from your project's real Azure DevOps states),

- an **"Add custom state..."** box (type any state name — useful for states the dropdown hasn't seen yet),
- **"Use Suggested"** (inserts a sensible default set for that stage),
- **"Clear"** (empties the stage).

You can mix both: start from "Use Suggested," then add or remove states to match your board exactly.

Stages you don't use → mark N/A

If your team doesn't have a stage — say you don't track *Test* or *Review* separately — choose **"— N/A (this stage is not used) —"** for it. N/A stages simply read **0** on the dashboards; they don't break your mapping and they don't raise "unmapped state" warnings. Every stage must contain **at least one state** *or* be explicitly **N/A**.

> 💡 The tool shows a live **"Your cycle time is calculated from these states"** preview as you edit, so you can see exactly which states start and stop the clock before you save.

---

5. Mapping suggestions by Azure DevOps process

Below are recommended starting mappings for the four standard Azure DevOps processes and for a rich custom board. Adjust to your team's reality — these are starting points, not rules.

> Legend: **➡ In Progress** is where the cycle-time clock starts. Anything in **New/Queued** is pre-work (excluded from cycle time). Anything you don't use → **N/A**.

Scrum process

Default states: `New, Approved, Committed, Done, Removed`

Stage	Map these states	Notes
New	`New`, `Approved`	Backlog — not counted.
Queued	`Committed`	*Only if work doesn't start immediately on commit.*
In Progress	*(none by default)*	**⬅ See note.**
Test	N/A	Add a custom state if you use one.
Review	N/A	Add a custom state if you use one.
Complete / Done	`Done`	
(cancelled)	`Removed` → leave unmapped	Treated as cancelled.

> **Scrum note — the most common question.** Out-of-the-box Scrum has no "in progress" state; work goes straight from `Committed` to `Done`. To get a meaningful cycle time you have two choices:
>
> 1. **Add a board state** like `In Progress` (recommended for richer flow metrics), then map it to **In Progress**; or
> 2. If `Committed` is effectively "*work has started,*" map `Committed` → **In Progress** instead of Queued. Cycle time then runs **committed → done**, which is usually what Scrum teams expect.

Agile process

Default states: `New, Active, Resolved, Closed, Removed`

Stage	Map these states
New	`New`
Queued	N/A *(or a custom "Ready/Committed" state)*
In Progress	`Active`
Test	N/A *(or a custom test state)*
Review	`Resolved`
Complete / Done	`Closed`
(cancelled)	`Removed` → leave unmapped

CMMI process

Default states: `Proposed, Active, Resolved, Closed, Removed`

Stage	Map these states
New	`Proposed`
Queued	N/A *(or a custom committed state)*
In Progress	`Active`
Test	N/A
Review	`Resolved`
Complete / Done	`Closed`
(cancelled)	`Removed` → leave unmapped

Basic process

Default states: `To Do, Doing, Done`

Stage	Map these states	Notes
New	`To Do`	Or split: keep backlog in New, committed items in Queued (if you distinguish them).
Queued	N/A	
In Progress	`Doing`	
Test	N/A	
Review	N/A	
Complete / Done	`Done`	

Rich custom board (recommended full setup)

If your team has invested in a detailed board, you get the richest analytics. This is the reference mapping we recommend for a full SDLC flow:

Stage	Map these states
New	`New`, `Approved`
Queued	`Committed`
In Progress	`In Progress`, `Active`
Test	`Ready To Test` *(dev done, PR merged, awaiting QA)*
Review	`Ready To Review`, `Ready To Accept`, `Resolved` *(QA done, awaiting product acceptance)*
Complete / Done	`Done`, `Ready To Release`, `Closed`
(cancelled)	`Removed` → leave unmapped

This mirrors a typical lifecycle: an item is **New** on the backlog → **Committed** into the sprint → developers move it to **In Progress** → on PR merge it becomes **Ready To Test** → after QA it's **Ready To Review / Ready To Accept** → once product accepts, it's **Done / Ready To Release / Closed**.

> **Don't have all these states?** That's fine — that's exactly what N/A is for. Map the states you *do* have, mark the rest N/A, and the dashboards adapt. You don't need to invent process to use the tool, and you don't need to abandon process you already have. You can also **create** these states on your board and have teams adopt them over time — the mapping makes either path work.

---

6. Special cases

- **Custom state names** — type them into the "Add custom state…" box on the relevant stage. Anything goes; the engine matches on your exact state names.
- **Non-English boards** — auto-detection recognises common state names in English, Spanish, Portuguese, Italian, German, French, Polish, Dutch and Swedish.
- **Multiple projects** — mappings are stored **per project**. Auto-detection seeds your first project; add the others from the same screen ("Add a project mapping").
- **Different work-item types** — you can give different item types (e.g. *Bug* vs *User Story*) their own mapping if their workflows differ. A general mapping applies to anything you haven't given a specific one.
- `Removed` **/ cancelled work** — see §3. Auto-detection excludes it; only map it to Done if you want removed items counted as completed.
- **Where it's saved** — the mapping is saved at the organisation level; an Azure DevOps **organisation admin** confirms and saves it. Once saved, it applies for everyone on your team.

---

7. Step-by-step

1. Open the extension and go to **Configuration** → **Workflow Mapping**.
2. Review the **auto-detected** mapping (or start fresh). Each of the six stages shows the states currently mapped to it.
3. For each stage, use the dropdown / **Use Suggested** / **Add custom state…** to match your board. Mark unused stages **N/A**.
4. Watch the live **"Your cycle time is calculated from these states"** preview — confirm the **start** states (In Progress / Test / Review) and **end** states (Done) are what you expect.
5. Resolve any **unmapped states** flagged in the banner (states the tool found on your items but you haven't placed).
6. Click **Save**. Your dashboards repopulate immediately.

---

8. Verifying it worked

- **Cycle Time card** shows a **Data Quality** chip (High / Medium / Low) based on how many completed items backed the numbers. Low usually means the mapping needs adjusting or there isn't enough completed history yet.
- If a chart is empty, the tool tells you *why*: "Workflow Mapping isn't set yet," "We couldn't find any completed items" (mapping likely needs adjusting), or "You're all set — waiting for completed work" (mapping is good, just no completed items yet).
- Spot-check a few recently completed items: their cycle time should match *first-active* → *done* on your board.

9. FAQ

Do I have to change how my team works? No. Map the states you already have. The tool adapts to your process, not the other way around.

What if we add or rename states later? Re-open Workflow Mapping, add/adjust the states, and save. New unmapped states are flagged automatically.

Why is my cycle time lower than I expected? Because time in **Queued** (committed-but-not-started) is excluded. If you want that time included, map your committed/working state to **In Progress** (see §5, Scrum note).

Can different teams have different mappings? Mappings are per project (and optionally per work-item type), so different projects can map differently.

Need a hand?

We're happy to walk through your specific board on a short call and get your mapping dialled in.

- 🎬 **Book a setup walkthrough** — https://ado-analytics.baytekdev.com/support/request
- 📖 **Docs** — https://ado-analytics.baytekdev.com/docs
- 🧭 **Features** — https://ado-analytics.baytekdev.com/features

Agile Analytics for Azure DevOps — https://ado-analytics.baytekdev.com