AZURE DEVOPS BACKLOG FIELD GUIDE

From strategy to shipping.

A text-first teaching guide for teams learning to build a backlog that is visible, understandable and honest.

Core idea: A backlog is not a collection of tickets. It is the story of why the team is spending its time, which product is affected, what outcome is being pursued, what must be delivered next and when the work is expected to happen.

Work becomes difficult to manage when its story disappears.

A request arrives in a meeting. A defect appears in production. A customer asks for an improvement. Someone notices an unsupported dependency. Before long, the team is carrying more work than anyone can reliably hold in memory.

When that work lives across conversations, email, personal notes and disconnected tickets, the team loses more than visibility. It loses the ability to make deliberate choices. People can see activity, but they cannot easily explain which work matters most, how it contributes to a larger outcome or what is competing for capacity.

> Azure DevOps gives the team a shared place to make the work visible. Its value does not come from filling in every field. Its value comes from helping the team tell a coherent story about the work.

That story begins with five questions:

1. **Why** are we doing this?
2. **Which product** does it belong to?
3. **What larger outcome and capability** are we pursuing?
4. **What is the next deliverable piece** of work?
5. **When** will we work on it and release it?

If work consumes meaningful team capacity, make it visible.

Every item starts with 'why?'

Before deciding whether something is an Epic, Feature or Milestone, first understand why the work deserves the team's time.

Some work advances an OKR

An Objective describes a meaningful direction. A Key Result gives the team a measurable sign of progress. If the organisation wants to rebuild customer trust and reduce customer-impacting incidents by 50%, reliability work can be connected directly to that result.

Some work protects or improves a KPI

A KPI is an ongoing measure of performance or health. Work that improves availability, response time or service quality may support a KPI even when it is not part of a time-bound OKR.

Some work supports neither - and still matters

Security patches, production incidents, technical debt, compliance, dependency upgrades, investigations and maintenance are legitimate uses of capacity. Record the honest reason rather than inventing a strategic relationship.

> If all work is forced into an OKR, leaders receive a distorted picture. Honest classification shows how much capacity advances strategy and how much protects, repairs or sustains the products.

Do not manufacture strategic alignment. Name the real reason for the work.

For this team, Area Path means Product.

Azure DevOps uses the term *Area Path*. That definition can feel abstract to a new team. We make it concrete by using one Area Path for each product the team manages.

If an item concerns Product B, select **Area Path = Product B**. That is all a team member needs to understand.

1	**Product A** Area Path All work owned by or delivered into Product A.
2	**Product B** Area Path All work owned by or delivered into Product B.
3	**Product C** Area Path All work owned by or delivered into Product C.
4	**Product D** Area Path All work owned by or delivered into Product D.

Area Path answers one question: Which product does this work belong to?

Resist adding sub-areas too early. Add another level only when the team has a genuine ownership or reporting problem that the extra structure will solve.

Move from the big outcome to the next deliverable.

The hierarchy connects day-to-day activity with the larger reason for doing it. Each level answers a different question.

1	**Reason** Why are we spending time? OKR, KPI, customer need, operational need, technical health, risk, compliance, incident or maintenance.
2	**Area Path** Which product? The product that owns or receives the work.
3	**Epic** What big outcome? A large initiative or substantial body of work, normally spanning several Features.
4	**Feature** What capability? A meaningful capability that can eventually be described as complete.
5	**User Story / Milestone** What do we deliver next? A smaller, understandable and testable increment that moves the Feature forward.

A note about 'Milestone'

Azure DevOps calls this work item a **User Story**. This team uses **Milestone** as everyday language for the same level. It is a local convention, not the standard industry definition, where milestone often means a significant checkpoint or event.

The hierarchy to remember: Why -> Product -> Epic -> Feature -> Milestone.

A story of customer trust.

Customers are experiencing repeated service interruptions. Trust is falling, support conversations are harder and some customers may leave. The organisation sets an Objective to rebuild customer trust and a Key Result to reduce customer-impacting incidents by 50%. Product B is the most exposed product.

WHY	Objective: Rebuild customer trust. Key Result: Reduce customer-impacting incidents by 50%.
AREA PATH	Product B
EPIC	Improve product reliability
FEATURE	Improve monitoring and alerting
MILESTONE	Alert support when API latency breaches the agreed threshold
ACCEPTANCE	The alert triggers at the correct threshold, reaches support and records enough information for investigation.
ITERATION	Q4 / Sprint 2
RELEASE	Version 5.2

A person who missed the original meeting can now see the strategic reason, affected product, larger outcome, capability, next deliverable, test for completion and expected timing.

When there is no OKR

WHY	An unsupported dependency creates security and operational risk.
AREA PATH	Product C
EPIC	Platform sustainability
FEATURE	Dependency modernisation
MILESTONE	Upgrade the framework from version X to version Y.
ACCEPTANCE	The build, security checks and agreed regression tests pass.

The backlog should describe reality, not merely the organisation's strategic intentions.

Iteration, Release and Delivery Plan tell different parts of the story.

Iteration Path: when are we working on it?

An Iteration is the sprint or time period in which the team intends to perform the work. An item can remain in the backlog without a specific Iteration until the team is ready to make a delivery commitment.

Release Version: when does it reach the product?

A Release identifies the product version expected to contain the completed change. The team may build and test work in Sprint 2 but ship it later in version 5.2. Iteration and Release are not interchangeable.

Delivery Plan: how does the wider picture fit together?

A Delivery Plan is not another work item. It is a timeline view across products, teams and Features. It helps reveal dependencies, competing priorities, timing problems and work that is taking longer than expected.

QUESTION	Azure DevOps concept
WHEN WILL WE WORK ON IT?	Iteration Path
WHICH PRODUCT RELEASE WILL CONTAIN IT?	Release Version
HOW DOES THE PORTFOLIO FIT OVER TIME?	Delivery Plan

A good backlog grows through consistent behaviour.

Make work visible

Capture Bugs, incidents, investigation, risk and maintenance when they consume meaningful capacity.

Capture before perfecting

A rough item is better than work hidden in a conversation. Refine it before committing it to delivery.

Build the parent chain

Milestones should normally have a Feature; Features should normally have an Epic.

Define done

Use acceptance criteria or another observable completion test.

Keep the backlog honest

Update priorities, dates and states when reality changes. Close work that will no longer happen.

Use Bugs consistently

A Bug is behaviour that differs from what was intended. A new capability is a Feature or Milestone.

> Beware ticket theatre. More fields do not automatically produce better decisions. Keep metadata only when the team will use it to prioritise, coordinate, learn or report.

Five questions before commitment

1. **Why?** What is the honest reason?
2. **Product?** Which Area Path owns it?
3. **Hierarchy?** What Epic and Feature provide context?
4. **When?** Is the team ready to assign an Iteration and intended Release?
5. **Done?** What observable conditions show completion?

The model at a glance.

QUESTION	USE	MEANING
Why are we doing it?	OKR, KPI or reason	The honest reason the work deserves capacity.
Which product?	Area Path	The product that owns or receives the work.
What big outcome?	Epic	A large initiative or substantial body of work.
What capability?	Feature	A meaningful capability that can be completed.
What next deliverable?	User Story / Milestone	A smaller, testable increment.
What is not working?	Bug	Behaviour that differs from what was intended.
When do we work on it?	Iteration Path	The sprint or delivery period.
When does it ship?	Release Version	The product version expected to contain it.
How does it all fit?	Delivery Plan	The wider timeline across products and teams.

What good looks like: Someone who was not in the meeting can understand what the team is doing, why it matters, which product it affects, what outcome it supports, what must happen next and roughly when it will be delivered.

REFERENCES

Official Microsoft guidance

Microsoft. (2026). *About areas and iteration paths.* Microsoft Learn. https://learn.microsoft.com/en-us/azure/devops/organizations/settings/about-areas-iterations

Microsoft. (2026). *About backlogs and boards.* Microsoft Learn. https://learn.microsoft.com/en-us/azure/devops/boards/backlogs/backlogs-overview

Microsoft. (2026). *Define features and epics to organise backlog items.* Microsoft Learn. https://learn.microsoft.com/en-us/azure/devops/boards/backlogs/define-features-epics

Microsoft. (2026). *Review team delivery plans.* Microsoft Learn. https://learn.microsoft.com/en-us/azure/devops/boards/plans/review-team-plans

Alternative view

Some organisations keep OKRs and KPIs outside Azure DevOps. This reduces administration and keeps the delivery system focused on execution. The trade-off is weaker traceability between strategic intent and the capacity the team actually uses. A sensible starting point is minimal strategic metadata, retained only when it supports real decisions.