Tell us about your PDF experience.

Azure Boards Configuration & Customization documentation

Configure and customize Azure Boards including area paths, iteration paths, projects, and processes.

Configure teams

CONCEPT

About teams & Agile tools

QUICKSTART

Configure team tools

Set team area paths

Set team iterations

Configure projects

CONCEPT

Configure & customize Boards

About projects & scaling up

QUICKSTART

Define area paths

Define iteration paths (sprints)

Customize an Inheritance process

OVERVIEW

Inheritance process model

QUICKSTART

Add a custom field

Add a custom work item type

TUTORIAL

Customize a workflow

Apply rules to workflow

Customize a project

Create & manage a process

Customize an On-premises XML process

OVERVIEW

On-premises XML process model

HOW-TO GUIDE

Add or modify a field

Add or modify a work item type

REFERENCE

XML element reference

Customize a process template (on-premises)

OVERVIEW

About processes & process templates

Process template files

HOW-TO GUIDE

Customize a process template

Upload/download process templates

REFERENCE

XML element reference

Reference

REFERENCE

Naming restrictions

Workflow states & categories

Work tracking object limits

Customize your work tracking experience

Article • 06/05/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

As you plan and track your project, consider configuring a feature or customizing your experience to align with your team's tracking requirements. The approach for customizing projects, which affects all teams, depends on the process model you're using.

This article gives you an overview of the customizations available and how they vary across the three process models. For specific guidance on customizations to support business decisions, Configure and customize Azure Boards. For more information, see What is Azure Boards? and About work items.

You can customize at the following levels of work tracking:

- **Project-level shared resources**: Define area and iteration paths which teams select to configure their backlogs and boards. Shared queries and work item tags are more objects that once defined can be shared across the project.
- **Team assets or tools**: Each team can configure their specific tools, such as backlogs, boards, and dashboards. For more information, see About teams and Agile tools.
- **Project and object-level permissions**: Manage access to work tracking tools, which include setting permissions for objects and the project and assigning users or groups to specific access levels.
- **Organization-level process customization**: Customize the fields, work item types, and backlogs and boards available to all teams.

Project-level shared resources

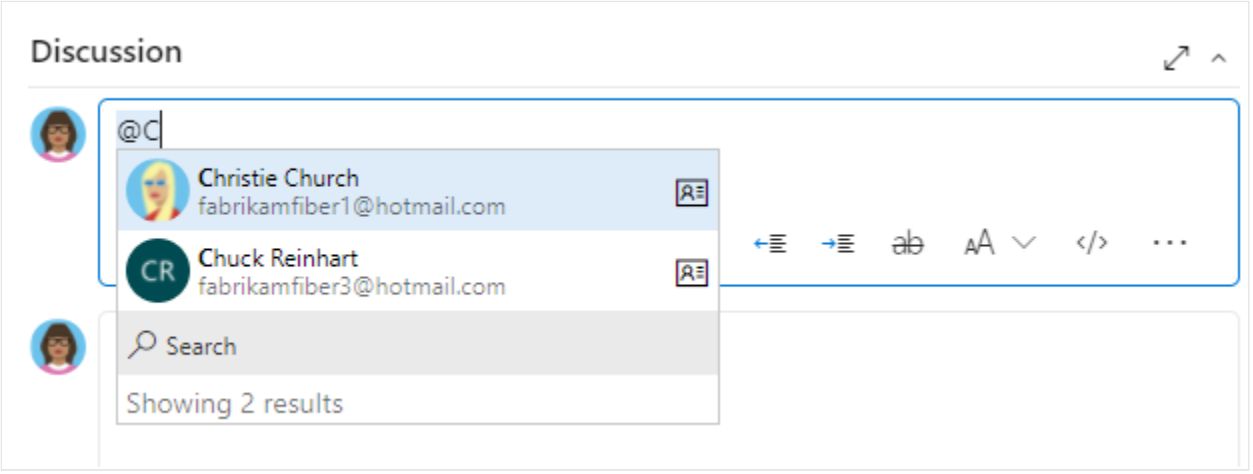
Each project provides many shared resources that support all teams within the project. You configure these features through the user interface or the admin context of the web portal. For more information, see the following articles:

- About area and iteration paths
- Set area paths
- Change the pick list for an iteration path
- Create and edit queries

- Add tags to work items

People picker and identity fields

- The **Assigned To** and other **Identity** fields are supported by the people picker feature.
- When you choose the **Assigned To** field within a work item form, the people picker is activated.
- To select a user, start entering their name and search until you find a match.
- Previously selected users appear automatically in the list.
- For organizations using Microsoft Entra ID or Active Directory, people pickers allow searching all users and groups added to the AD (not just ones added to a specific project).
- To limit the scope of identities available for selection to project-specific users, use the **Project-Scoped Users** group.
- Custom rules can further restrict the values available for Identity fields within a work item.



For more information, see the following articles:

- Add Active Directory / Microsoft Entra users or groups to a built-in security group.
- Limit identity search.

Organization-level process customization

Your project defines the work item types (WITs) available for tracking work and configures Agile tools. It specifies user stories, tasks, bugs, and the data fields used to capture information. Customized objects are shared across teams within the project.

> ⓘ Note

The method you use to customize work tracking depends on the process model you subscribe to:

- **Inheritance**: Supports WYSIWYG customization, available for Azure DevOps Services, Azure DevOps Server 2019, and Azure DevOps Server 2020.
- **Hosted XML**: Supports customization through import/export of process templates, available for a select number of customers of Azure DevOps Services who have opted into this model.
- **On-premises XML**: Supports customization through import/export of XML definition files for work tracking objects and is available for all on-premises deployments.

The following table summarizes the differences between the three supported process models. For definitions of the main work tracking objects, see Agile glossary. For links to customization articles, see Quick reference index for Azure Boards settings.

Feature	Inheritance	Hosted XML	On-premises XML
WYSIWYG editing	✔		
Create inherited custom processes, Inherit changes in system processes (Agile, Basic, Scrum, CMMI)	✔		
Create custom process templates (see note 1)	✔	✔	
Updated process changes automatically apply to all projects referencing the process	✔		



Support for customizing fields, work item types, form layout, workflow, custom rules, backlog levels, custom controls, test management







Support for customizing link types, team fields, global workflow, and process configuration (see note 3)



Initial configuration of Area paths, Iteration Paths, work item queries, security groups, and permissions (see note 3)





Global lists

Picklists

(see note 2)



Use `az boards` command-line tools to edit projects and teams and list information







Use the `witadmin` command-line tools to list and export process information







REST API (read)







REST API (write)





(see note 5)

Notes:

1. A process determines the building blocks used to track work. A process template specifies an interdependent-related set of XML definition files that provide the building blocks and initial configuration for tracking work and other functional areas.
2. Hosted XML customization supports adding and updating global lists with a process update (subject to limits on maximum size of each list). For more information, see Work tracking object limits.
3. The Inherited process model doesn't support customization of the following features available with customization of process templates. Instead, you customize these areas within the web portal on a project-by-project basis.

   - Area and iteration paths
   - Work item queries
   - Security groups and permissions
   - Permissions and access to functional areas such as version control and build

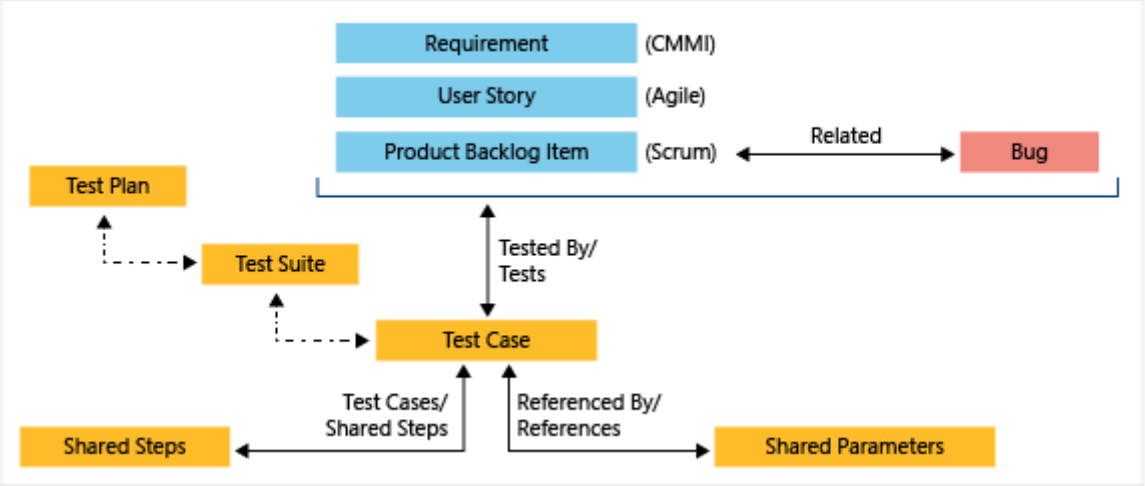
   Or, you can use REST APIs or the Azure DevOps CLI command tool.
4. Support for Office Project integration with Azure DevOps is deprecated and the `TFSFieldMapping` command isn't supported.
5. Use the REST API to import and export process templates.

Customize the test experience

Several work item types support the test experience within the web portal **Test** pages and Test Manager client.

- For an **Inherited process**, you can customize the following work item types as you would any other work item type:
  - Test Plan
  - Test Suite
  - Test Case
- For an **On-premises XML process**, you can customize all test-related work item types, including:
  - Test Plan
  - Test Suite
  - Test Case
  - Shared Steps
  - Shared Parameters

The following example shows the supported link relationships.



Less common customizations

You can only perform the following customizations when working with the Hosted XML or On-premises XML process models. Customizations made to process configuration apply to all teams within a project.

Backlog and board limits (Hosted XML, On-premises XML)

To limit the display load time to acceptable parameters, the task board is restricted to a maximum of 1,000 work items. For details, see Process configuration XML element reference.

You can increase this value up to a maximum of 1500 by specifying a value for the `workItemCountLimit` attribute of the **TaskBacklog** element. For details, see Process configuration XML element reference.

```
<TaskBacklog category="Microsoft.TaskCategory" pluralName="Tasks"
singularName="Task" workItemCountLimit="800" >
    . . .
</TaskBacklog>
```

Change field assignments (Hosted XML, On-premises XML)

You can change the work item fields that are used in calculating capacity, burndown charts, forecasting, and velocity. Any change you make to one of the default assignments should correspond to a change made to the WIT used to define and capture information for that value.

For example, if you change the `refname` assigned to `type="Activity"` then you should include the same field in the WIT definition assigned to the Task Category that captures the activity information. For details, see Process configuration XML element reference.

The fields you assign are used by the following tools:

Expand table

Tool	Field type
Task board, capacity tools, sprint burndown	Remaining work
Product and portfolio backlogs	Backlog priority
Velocity and forecast	Effort (maps to Story Points, Effort, or Size)
Task board, capacity tools	Remaining work
Capacity tools	Activity (Task Activity or Discipline)

Manage access to work tracking tools

Manage access to specific features through permission settings. When you add user accounts to your team, they're automatically added to the Contributor group. They then have access to most of the features they'll need to contribute to code, work tracking, builds, and test. However, the Contributor group doesn't allow users to create shared

queries or to add area or iteration paths. You have to grant these permissions separately.

You can manage access with the following permission settings:

- When you add user accounts to your team, they're automatically added to the **Contributor** group.
- The **Contributor** group provides access to most features needed for contributing to code, work tracking, builds, and testing.
- But, the **Contributor** group doesn't allow users to:
  - Create shared queries
  - Add area or iteration paths
  - To grant these permissions separately, follow the appropriate steps.
- For a simplified overview of common default permissions and access assignments, see Permissions and access. If you're new to managing permissions, explore Get started with permissions, access, and security groups, Permission inheritance and security groups.

To manage access to specific features, see the following articles:

Manage access

- About access levels
- Add team members (Azure DevOps Services)
- Change access levels (on-premises deployments)
- Add team members (on-premises deployments)

Permissions

- Area path permissions
- Process permissions
- Work item query and folder permissions
- Dashboard permissions
- Delivery Plan permissions
- Tagging permissions
- Test permissions

Shared resources

- Alerts
- Area paths
- Iteration paths
- Queries
- Tags

More customization options

Choose from the following other customization options:

- Check out Marketplace extensions to see if there's a tool available for your purposes
- Develop your own extension
- Determine if a Service hook satisfies your needs
- Create your own tool using REST APIs
- Add a feature request to our Developer Community page.

Next steps

Configure and customize Azure Boards

Related articles

- Work item field index
- witAdmin command-line tool
- Manage process templates
- Process templates overview
- Work tracking, process, and project limits

Feedback

Was this page helpful? Yes No

Provide product feedback

Quick reference index for Azure Boards settings

Article • 03/25/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Use this index to quickly access concepts and tasks related to configuring and customizing Azure Boards. If you're new to the Azure Boards, see What is Azure Boards?.

Get started

You can start using Azure Boards without configuring or customizing anything. However, as your organization grows, you'll find it beneficial to add teams, define area paths and iteration paths. If you use GitHub for your source code repository, you may want to connect Azure Boards to GitHub. If you are tasked with administrating a project for several teams, we recommend you review the following articles.

- Customize work tracking overview
- Add a team
- Define Area Paths
- Define Iteration Paths
- Configure and customize Azure Boards
- Connect Azure Boards to GitHub
- Set process permissions

Configure team tools

Each team administrator can configure the following elements for their team, backlogs, or boards. For an introduction to configuring team tools, see Manage and configure team tools.

General

- Define Area Paths for a team
- Define Iteration Paths for a team
- Define work item templates
- Backlogs
- Working with bugs

Kanban boards

- Columns
- WIP limits
- Definition of Done
- Split columns
- Swimlanes
- Card fields, styles, tag colors, annotations, and card reordering

Process and project customization

You customize your work tracking experience to support your business and reporting needs. The most common customizations include adding a custom field, modifying a work item form, or adding a custom work item type. Process customization tasks must be performed by an administrator who has the necessary organization-level permissions, as they affect all users and teams within a project.

You customize Azure Boards work tracking using the Inheritance process model which uses a WYSIWYG user interface to customize work tracking objects and tools. For select organizations who migrated to Azure DevOps Services using the data migration tool for Azure DevOps, the Hosted XML process model is used. This model relies on modifying and importing process templates to customize work tracking objects and tools.

Inheritance process model

With the Inheritance process, you apply a process to a project.

- Create and manage inherited processes
- Customize a project using an inherited process
- Change the reference process from Agile to Scrum
- Change the reference process from Basic to Agile
- Change the reference process from Scrum to Agile

The Inheritance process model supports the following customization tasks.

Fields

- Add a checkbox (Boolean) field
- Add/remove custom fields
- Add/remove custom rules to a field
- Add a person-name/Identity
- Add a picklist (drop-down menu)
- Add a rich-text (HTML) field
- Change a field label

- Delete field

Work item types

- Add a custom work item type
- Add/remove custom fields
- Add/remove custom groups
- Add/remove custom pages
- Add/remove a custom control
- Add custom rules
- Change the color or description
- Delete a work item type
- Enable/disable a work item type

Workflow customizations

- Hide or unhide a state
- Add, edit, or remove a workflow state
- Apply rules to workflow states

Backlogs and boards

- Add a custom work item type to a board
- Add an inherited work item type
- Change the default work item type
- Rename a backlog
- Add a portfolio backlog which displays custom work item types
- Edit or rename a portfolio backlog
- Delete the top-level custom portfolio backlog

Hosted XML process model

With the Hosted XML process model, you customize work tracking objects and Agile tools by modifying and importing a process template. Updates made to the process template are applied to projects that were created using that process.

> ⓘ **Note**
>
> The Hosted XML process model is only supported for organizations that have migrated to Azure DevOps using the data migration tool for Azure DevOps.

Fields

- Add or modify a field
- Add a checkbox (Boolean) field

- Add rules to a field
- Change a field label
- Add a custom control field
- Remove a field
- Define global lists

Work item types

- Add or modify a work item type
- Customize the form
- Specify the work item type color
- Specify the work item type icon
- Customize the workflow (States, Reasons, Transitions)
- Specify the workflow state color

Backlogs and boards

- Add WITs to backlogs or boards
- Add portfolio backlogs
- Configure the quick add panel
- Configure the default backlog columns
- Set maximum number of task board items
- Set default weekend days (Scrum)
- Set default bug management behavior
- Set default hidden backlogs
- Change process configuration
- Add custom categories

Process templates

- Import a process template
- Export a process template
- Customize a process template

Related articles

- Work item field index
- witadmin command-line tool
- Naming restrictions and conventions
- Work tracking, process, and project limits

Feedback

Was this page helpful? Yes No

Provide product feedback

Area and iteration paths

Article • 11/04/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

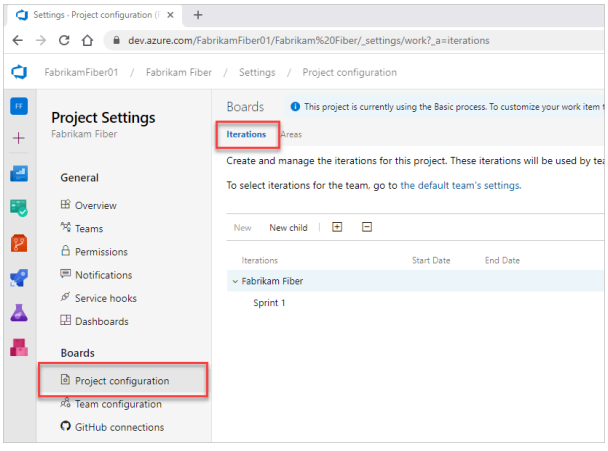
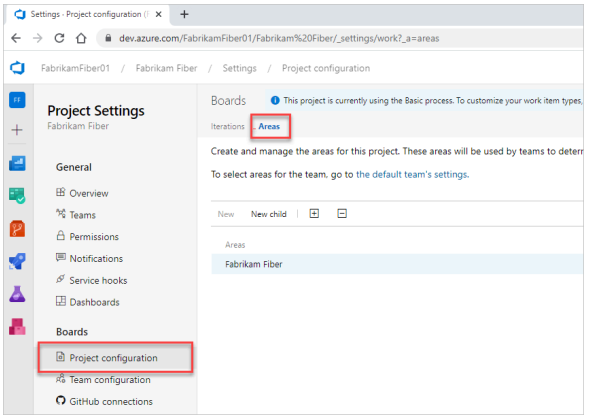
Area paths group work items by team, product, or feature area. Iteration paths group work into sprints, milestones, or other time-related periods. Both fields support hierarchical paths.

Define area and iteration paths for a project, and teams can select which paths to use for their backlog and Agile tools. Learn how Agile tools use these paths in Agile tools that rely on areas and iterations.

> **Note**
>
> Area paths and iteration paths are also known as *Classification Nodes*. You can manage them programmatically using the Classification Nodes (REST API) or the Azure DevOps CLI command az boards iteration.

The areas and iterations depend on the process used to create your project. This example shows the default settings for the Scrum process. Dates aren't set by default; you need to set dates to match your sprint or release schedules.

Expand table

Iterations	Areas
	

> **Important**

- Deleting **Area Paths** or reconfiguring **Iteration Paths** causes irreversible data loss. For example, burndown and burnup widget charts, sprint burndown, and velocity charts for teams with changed **Area Paths** won't display accurate data. Historical trend charts reference the **Area Path** and **Iteration Path** as they were defined at the time for each work item. Once deleted, you can't retrieve historical data for these paths.
- You can only delete area and iteration paths that are no longer used by any work items.

Define and assign area paths

If you're new to managing projects and teams, follow these steps to configure your project and teams:

1. **Determine area paths**: Identify the number and names of **Area Paths** needed to categorize your work. At a minimum, add one area path for each team you define.
2. **Decide on teams**: Decide on the number and names of teams you want to support. For more information, see About teams and Agile tools.
3. **Define area paths**: Define the area paths to support steps 1 and 2 at the project level. Follow these steps: Add area paths.
4. **Define teams**: Define the teams needed to support step 2. For more information, see Add a team, move from one default team to several teams.
5. **Configure team settings**: Use these steps to assign the default and other area paths to each team.
6. **Assign work items**: Assign work items to the area paths you defined. Use bulk modify to modify several work items at once.

> ⓘ **Note**
>
> You can define up to 10,000 **Area Paths** per project and assign up to 300 **Area Paths** to a single team. For more information, see Work tracking, process, and project limits.
>
> You can assign the same **Area Path** to more than one team, but this can cause problems if two teams claim ownership over the same set of work items. For more information, see Limitations of multi-team board views.

You can do the following actions at any time:

- Add more child nodes
- Rename an area path (except the root area path)
- Move a child node under another node
- Delete a child node
- Rename a team
- Change the area path assignments made to a team

For more information, see Configure a hierarchy of teams.

How many areas should a team define?

Add areas to support your team's traceability and security requirements. Use areas to represent logical or physical components, and create child areas to represent specific features.

Add areas when you need to do any of the following tasks:

- Filter queries based on a product or feature area
- Organize or group work items by team or subteams
- Restrict access to work items based on their area

Each team can create a hierarchy of areas to organize their backlog items, user stories, requirements, tasks, and bugs.

Avoid creating an overly complex area structure. While you can use areas to partition permissions on work items, complex trees require significant overhead for permission management. Duplicating the structure and permissions in other projects can become cumbersome.

Define and assign iteration paths

Follow these steps to configure **Iteration Paths** for your project and teams:

1. Define the area paths and teams using the guidance in Define area paths and assign to a team.
2. Determine the length of the iteration you want to support. We recommend that all teams use the same sprint cadence.
3. Decide whether you want a flat structure or a hierarchy of sprints and releases.
4. Define the iteration paths to support steps 2 and 3 at the project level. Follow these steps: Add iterations and set iteration dates.
5. Open the team configuration and assign the default, backlog, and other iteration paths to each team. Follow these steps: Open team settings and Set team default

iteration paths.

6. Each team should assign an iteration path to their work items that falls under the **Backlog iteration** path. These work items appear on their product backlogs and boards. Use bulk modify to modify several work items at once. See also Assign backlog items to a sprint.

> ⓘ **Note**
>
> You can define up to 10,000 **Iteration Paths** per project and assign up to 300 **Iteration Paths** to a single team. For more information, see Work tracking, process, and project limits.

You can do the following actions at any time:

- Add more child iteration nodes
- Rename an iteration path (except the root path)
- Move a child iteration path under another node
- Delete a child iteration path
- Change the default and selected iteration paths assigned to a team

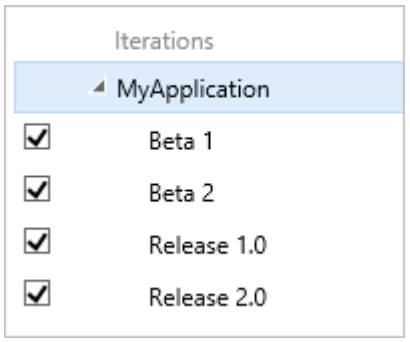
How many iterations should a team define?

Define as many child iterations as needed to reflect your project lifecycle. These paths represent a series of events, such as sprints, prebeta and beta results, and other release milestones. Teams usually leave work items assigned to the team's default iteration if they aren't yet scheduled for work or release. Define as many child iterations as needed to reflect your project lifecycle. These iterations can represent various events, such as sprints, prebeta and beta phases, and other release milestones. Teams typically leave work items assigned to the team's default iteration if they aren't yet scheduled for work or release.

Add iterations to support the following requirements:

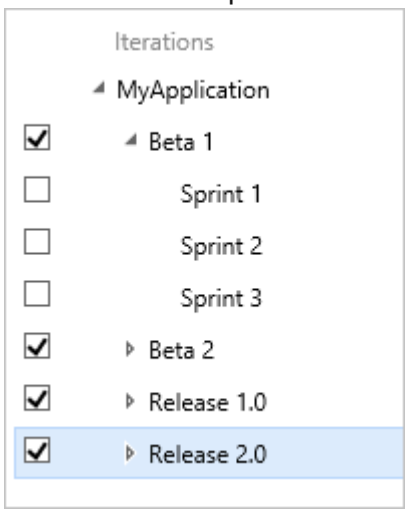
- Define sprints for your Scrum teams to plan and execute their sprints
- Set up more complex multi-release and sprint cycles
- Filter queries based on sprints, milestones, or cycle time for your project
- Support future work that isn't ready to be assigned to a target release cycle.

In the following example, Beta 1, Beta 2, Release 1.0, and Release 2.0 are defined for the MyApplication project.



As you create the backlog of product features and tasks, assign them to milestones based on when you expect the team to complete them. As your needs change, you can add events under each major milestone to reflect how your team schedules and manages its work.

For example, the Beta 1 iteration now contains three child nodes, one for each sprint in the Beta 1 time period.



Iterations don't enforce any rules. For example, you can assign a task to an iteration without closing or completing it during that iteration. At the end of an iteration, identify all work items that remain active or open and take appropriate action. You can move them to a different iteration or return them to the backlog.

You can run queries to find features and work items that are assigned to a specific iteration or a set of iterations and then bulk modify the work items to change their iteration paths. For more information, see Query by date or current iteration

Naming restrictions

The **Area Path** and **Iteration Path** fields, data type=TreePath, consist of multiple node items separated by the backslash (\) character. Minimize the names of nodes and make sure you conform to the following restrictions when you're adding child nodes.

Expand table

Restriction type	Restriction
Node length	Must not contain more than 255 characters.
Reserved names	- Must not consist only of a period `.` or two periods `..`. - Must not be a system-reserved name, such as PRN, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, COM10, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9, NUL, CON, or AUX. For more information about reserved names, see File Names, Paths, and Namespaces.
Special characters for nodes	- Must not contain Unicode control characters. - Must not contain any one of the following characters: `\ / : * ? " < > | # $ & * +`. - Must not contain characters prohibited by the local file system. For more information about Windows character restrictions, see Naming Files, Paths, and Namespaces.
Path length	Must not contain more than 4,000 Unicode characters.
Path hierarchy depth	Must be fewer than 14 levels deep.

Related articles

- Define area paths and assign to a team
- Define iteration paths and configure team iterations
- Use Agile tools and define sprints
- Query by date or current iteration

ⓘ **Note:** The author created this article with assistance from AI. Learn more

Feedback

Was this page helpful? Yes No

Provide product feedback

Define area paths and assign to a team

Article • 11/14/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Add area paths to organize teams and group work items by product, feature, or business areas. Assigning area paths to teams ensures that work items under those paths appear on the respective team's backlogs and boards. By default, all work items are assigned to the root area path and appear in the project's default team.

After you define area paths at the project level, assign them to teams within the team configuration. You can also create a hierarchical structure of area paths to support subareas, with up to 14 levels deep.

To perform the following tasks, you must define area paths:

- Query and chart work items based on Area Path
- Assign work to more than one team
- Work with management and feature teams
- Filter a backlog, query, board, or plan using Area Paths

> **Tip**
>
> You can define your area path structure and assign area paths to teams. Or, you can add a team and create the area path with the team name at that time. If teams are fully independent, create a flat set of area paths. However, if you want to create a hierarchy of teams, then you'll want to create a tree-hierarchy of area paths. For more information, see **Configure a hierarchy of teams**.

To use the following tools, teams must subscribe to area paths:

- Boards> all tools
- Backlogs> all tools
- Sprints> all tools
- Delivery plans
- Rollup
- Analytics> all

To understand how the system uses area paths, see About area and iteration paths.

Prerequisites

Permissions:

- To set team area paths, be a team administrator or a member of the Project Administrators group. By default, the user who created the project has these permissions.
- To add an area path under the root node or edit or delete any child node, be a member of the Project Administrators group.
- To add, edit, and manage area paths under a node, have one or more of the following permissions set to **Allow**:
  - **Create child nodes**
  - **Delete this node**
  - **Edit this node**
  - **View permissions in this node**

For more information, see Set permissions and access for work tracking.

For naming restrictions on area paths, see About areas and iterations, Naming restrictions.

Get started

Each team has access to various Agile tools, as described in About teams and Agile tools. These tools reference the team's default area paths. Most teams use one area path and several iteration paths for work tracking, but teams can also use multiple area paths on their backlogs and boards to support different scenarios.

New projects include a single root area corresponding to the project name and a default team with the same name assigned to this root area path.

If you're new to managing projects and teams, follow these steps to configure your project and teams:

1. **Define area paths**: Determine the number and names of area paths needed to categorize your work. Add at least one area path per team. For more information, see About areas and iterations.
2. **Define teams**: Decide the number and names of teams required. For more information, see About teams and Agile tools.
3. **Configure area paths**: Go to **Project settings** > **Project configuration** > **Areas** and define the area paths for your project. Follow the steps in Add area paths.
4. **Create teams**: Create the necessary teams based on step 2. For more information, see Add a team, move from one default team to several teams.

5. **Assign area paths to teams**: In the team configuration, assign the default and other area paths to each team. Do the steps in Set team default area paths.
6. **Assign area paths to work items**: Assign work items to the defined area paths using bulk modify to update multiple items at once.

> **Note**
>
> Assigning the same area path to multiple teams may cause conflicts over work item ownership. For more information, see Limitations of multi-team board views.

As needed, you can do the following actions at any time:

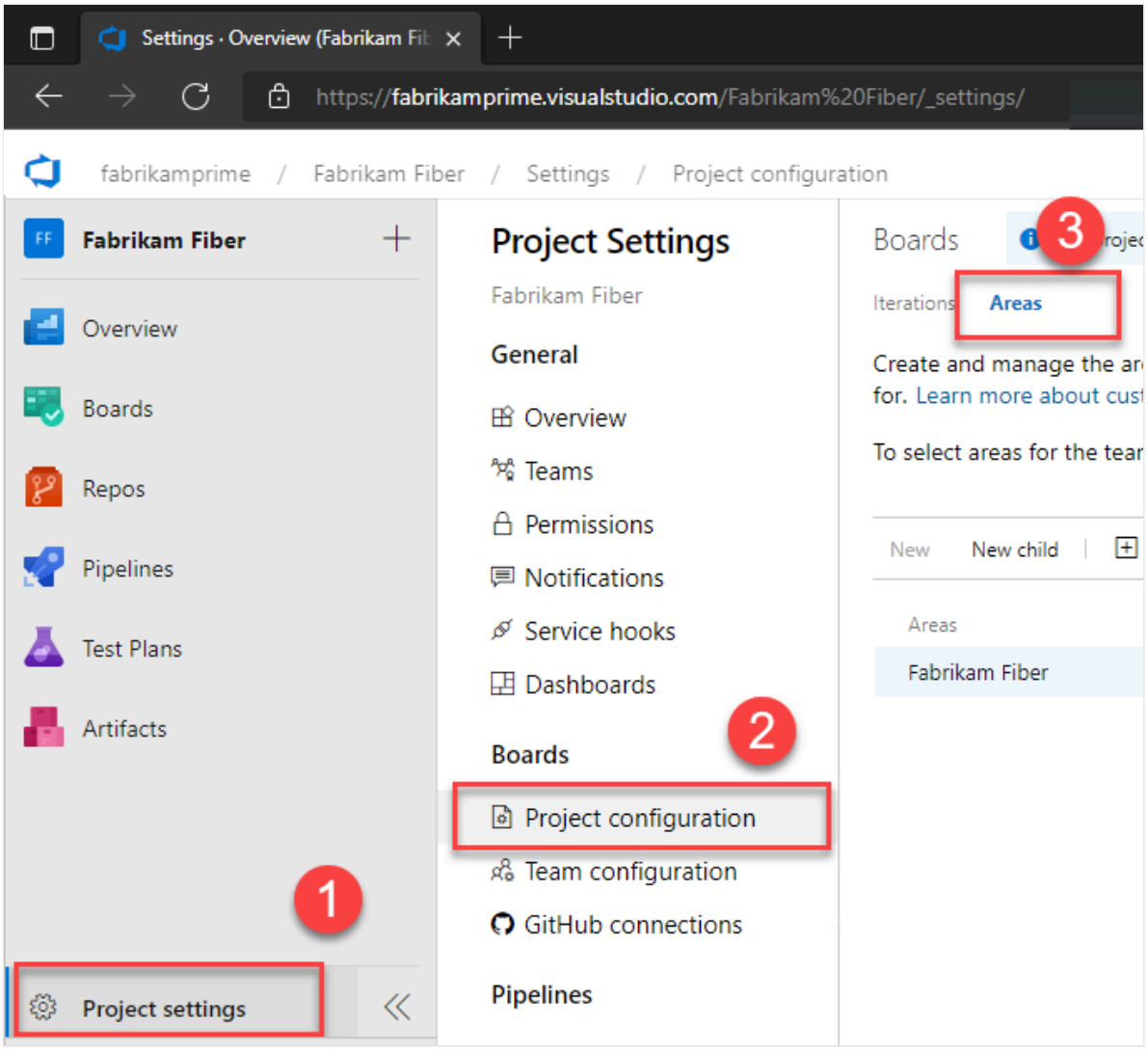
- Add other child nodes
- Rename an area path (except the root area path)
- Move a child node under another node
- Delete a child node
- Rename a team
- Change the area path assignments made to a team

List project areas

To list project areas, do the following steps:

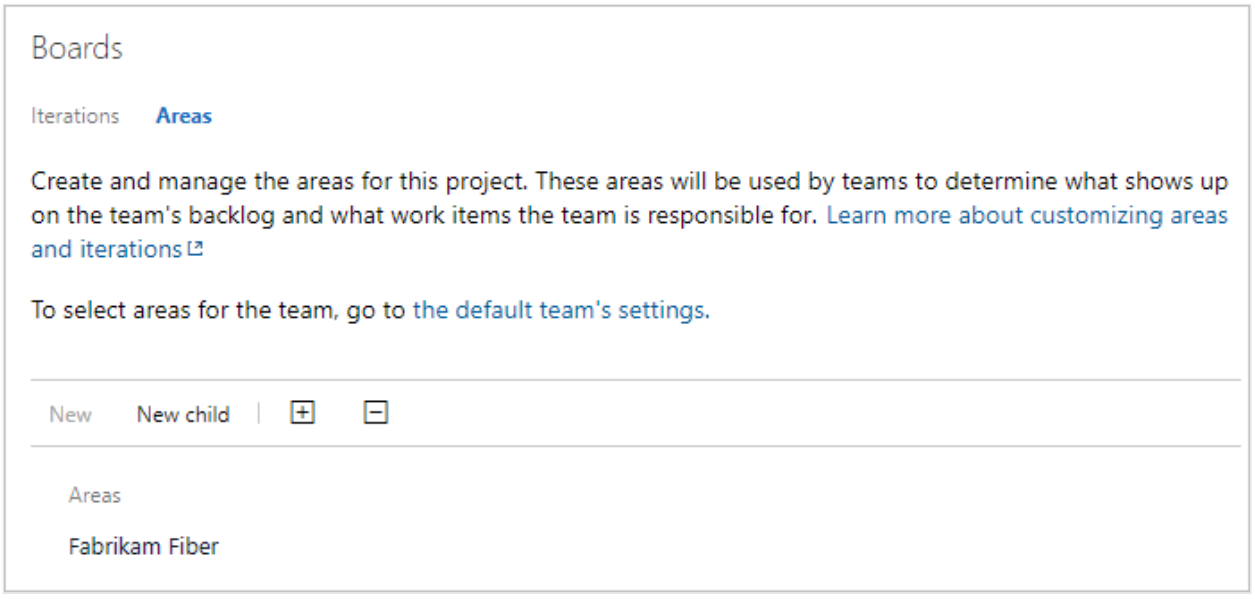
Browser

From your web portal, select (1) **Project settings** > (2) **Project configuration** > (3) **Areas**.



Add an area path

Only one root area is defined by default. Add new and child area paths under the root area path for the project, or as a child to another area path or team. New area paths are added to the root node by default.



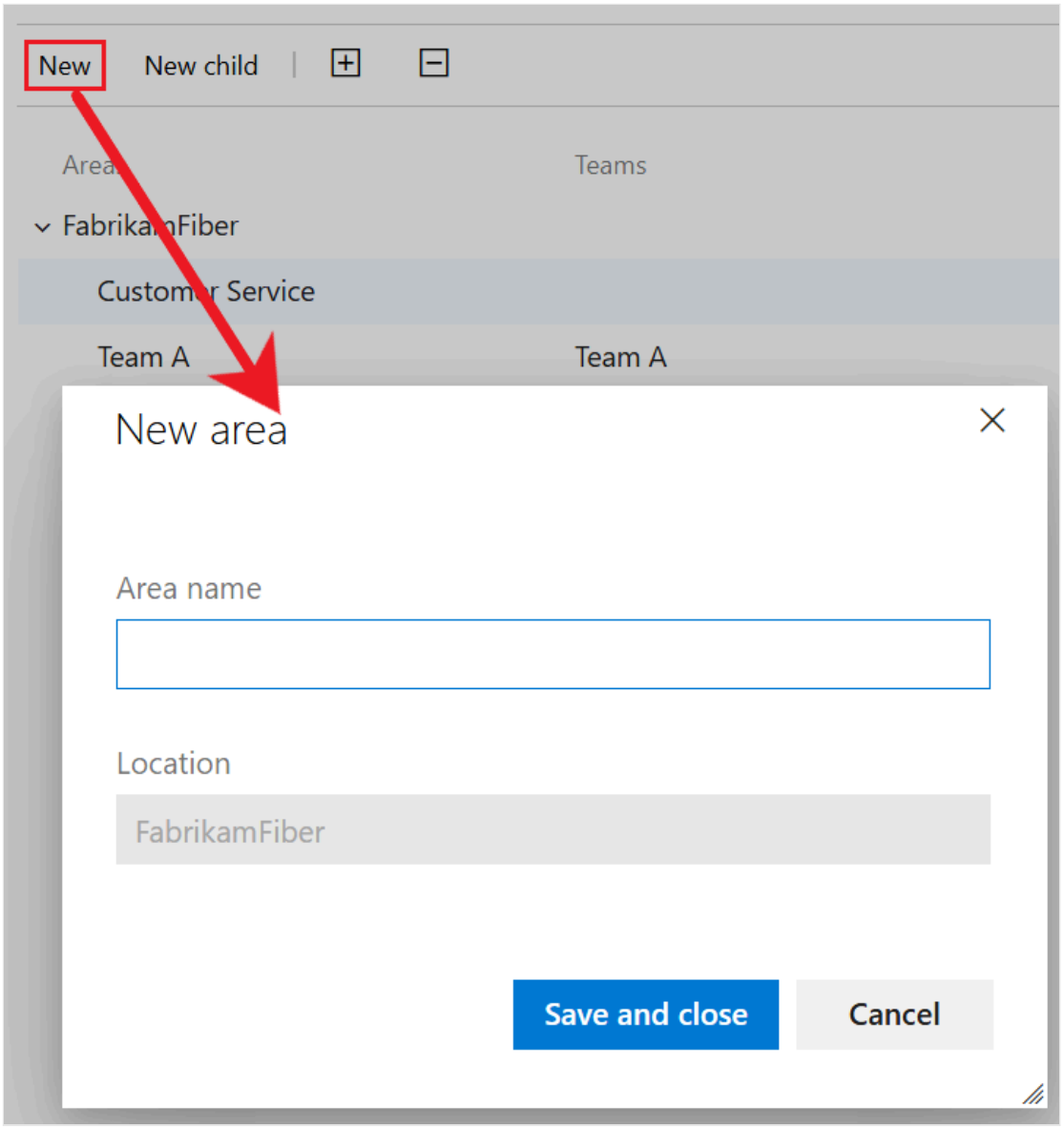
You can also add new and child area paths to a team. For more information, see Set team area paths.

Browser

To add a new area path, do the following steps:

1. Highlight the area path and then choose **New child**. Optionally, you can select ··· for an area path and choose **New**, or select the **New** menu item at the top of the area list.

2. Enter a name (255 characters or less) for the node. For more name restrictions, see About areas and iterations, Naming restrictions.



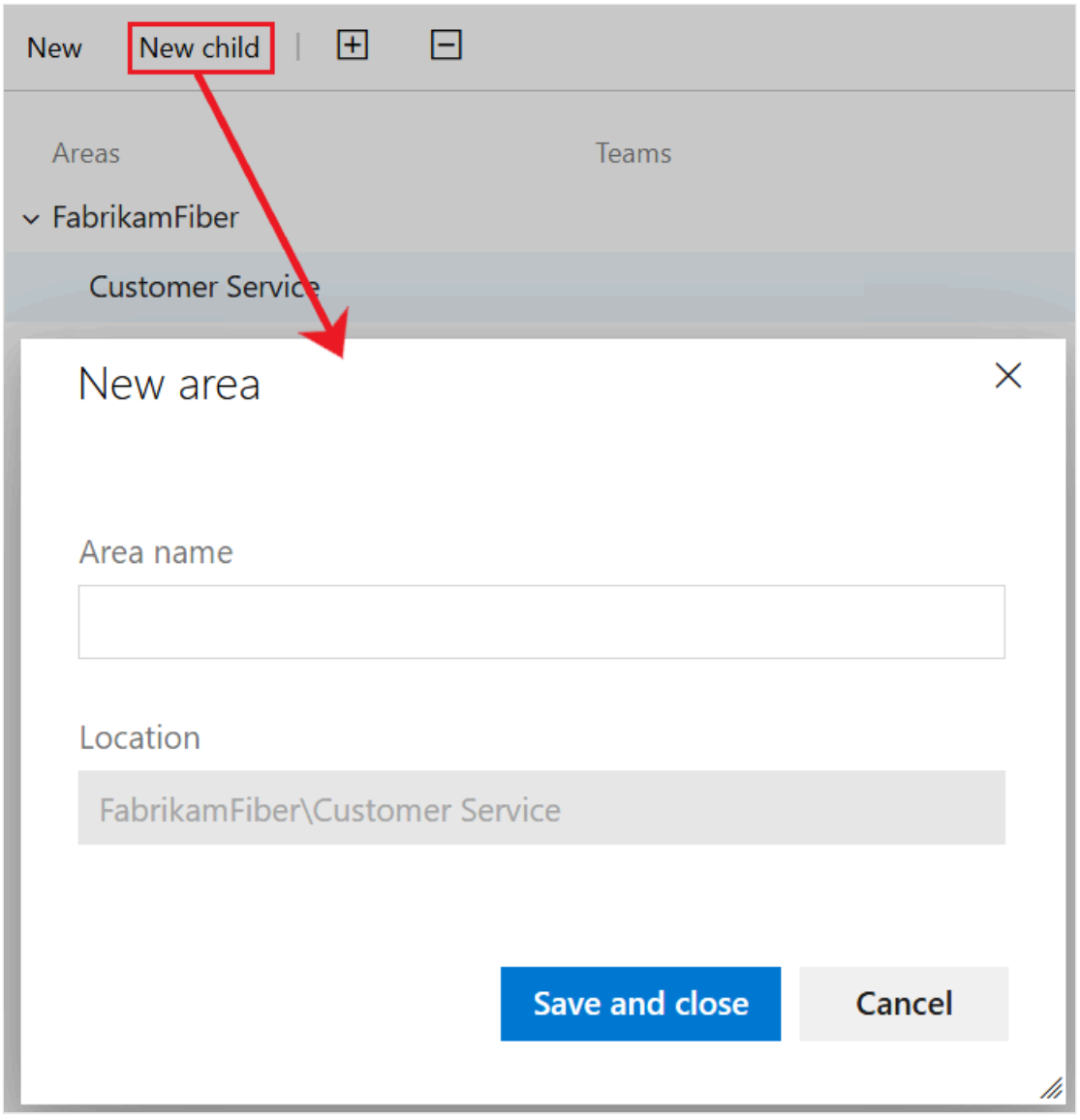
3. Select **Save and close**

All new areas are added as a child node to the root area path.

To add a child area path, do the following steps:

1. Highlight the area path and then choose **New child**. Optionally, you can select ··· for the area path and choose **New child** or select **New child** from the menu at the top of the area list.

2. Enter a name (255 characters or less) for the node. For more name restrictions, see About areas and iterations, Naming restrictions.



3. Select **Save and close**

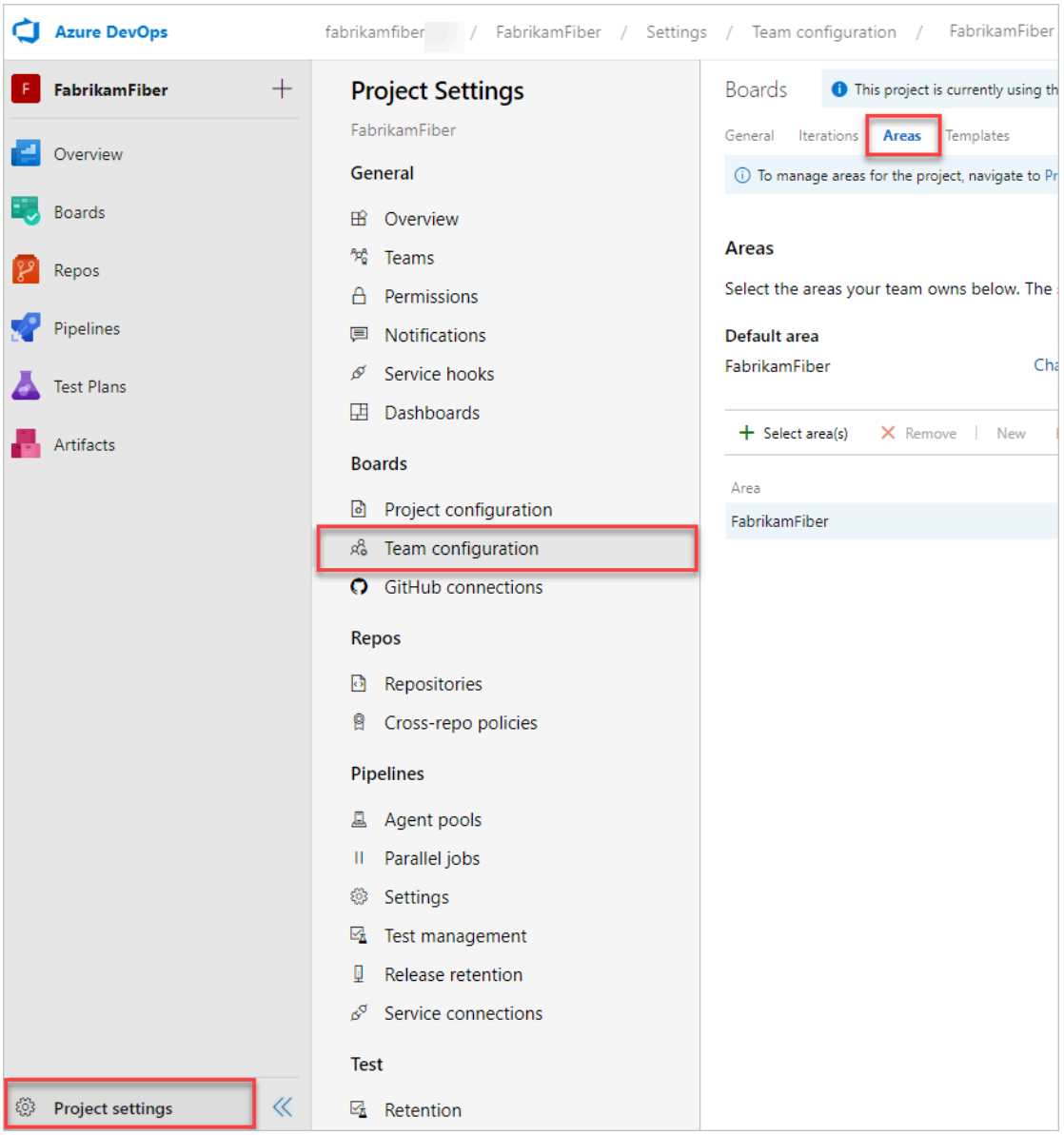
You can also add area paths to a team on the **Team configuration** > **Areas** page.

List team area paths

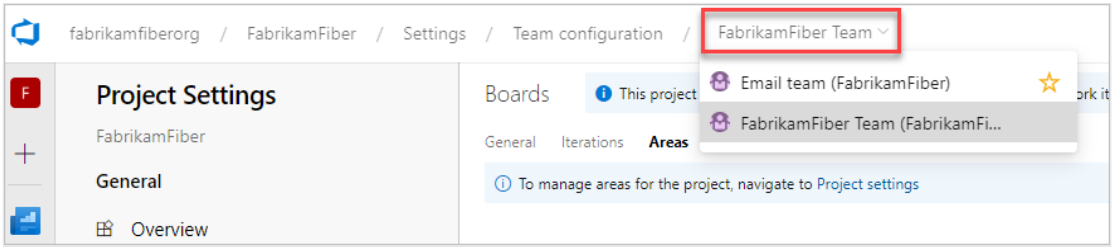
You set team defaults from team settings. If you're not a team administrator, get added as one. Only team or project administrators can change team settings.

Browser

1. Open your project, and then select **Project settings** > **Team configuration** > **Areas**.



2. If you need to switch the team context, use the team selector within the breadcrumbs.

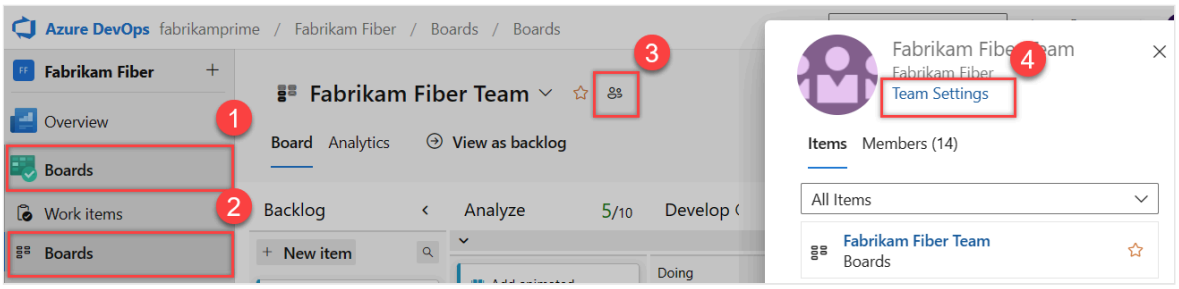


Open team settings from the team profile

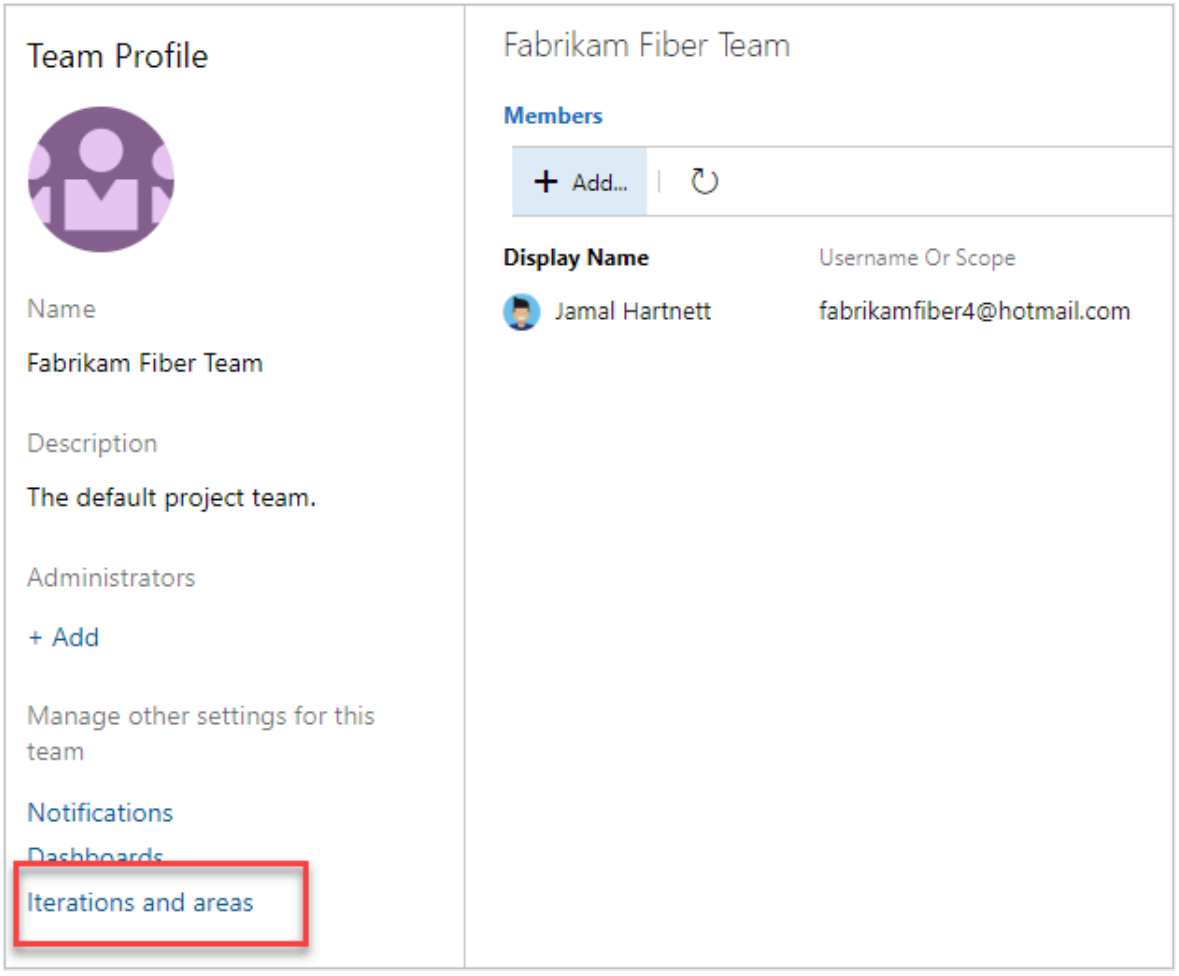
You define both areas and iterations from **Project Settings** > **Team configuration**. You can quickly navigate to it from a team work tracking backlog, board, or dashboard.

1. Open a backlog or board for a team and choose **Team profile** > **Team Settings**.

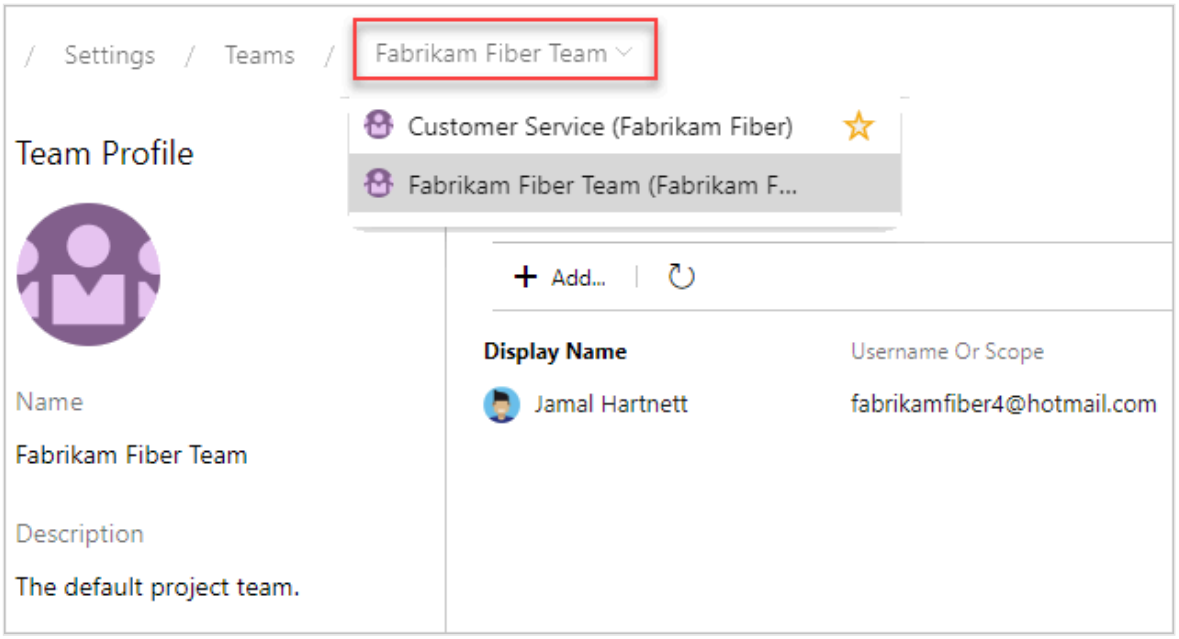
   Here we open the Board for the Fabrikam Fiber team and from there the team profile.



2. Choose **Iterations and areas**.



3. If you need to switch the team context, use the team selector within the breadcrumbs.



Set team area paths

All work items that are assigned to a team area path appear on the backlogs and boards for that team. You can select one or more area paths and optionally include their subarea paths. Choose to include subarea paths when you want to support rollup views of work done across several teams or areas.

> **Note**
>
> Teams can be assigned a maximum of 300 **Area Paths**. For more information, see Work tracking, process, and project limits.

The default area path determines the default area path assigned to work items that are created from the team context.

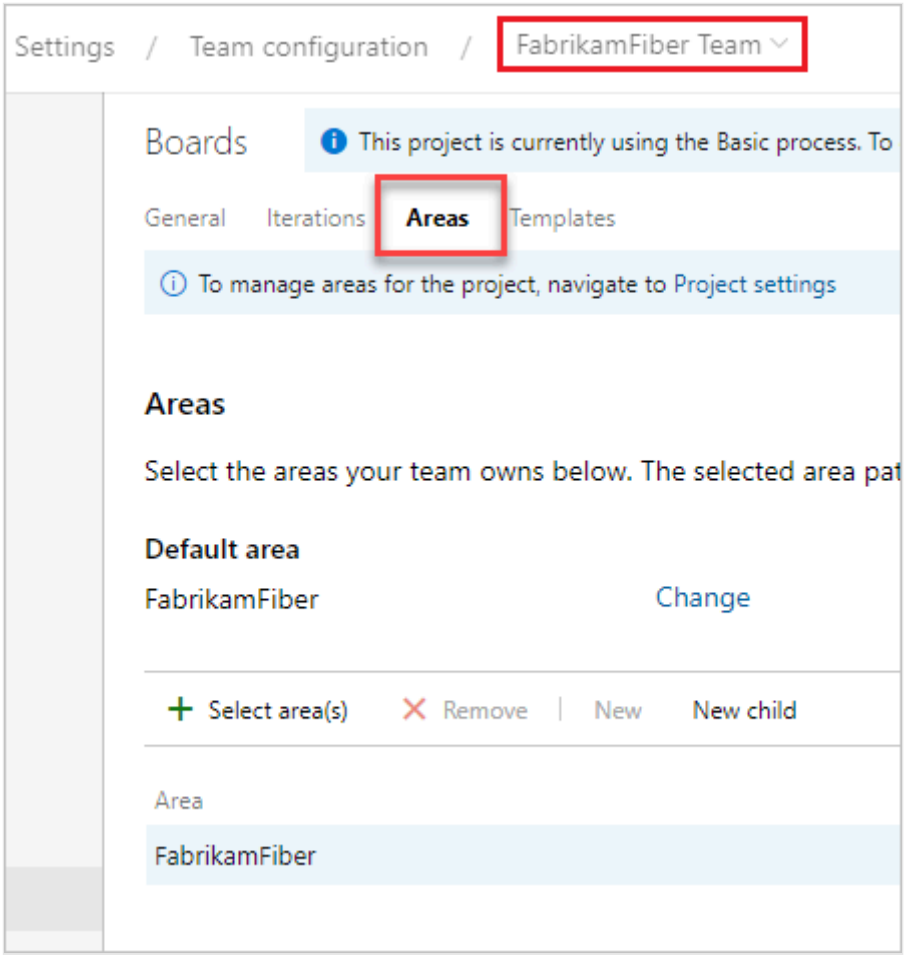
> **Important**
>
> Work items that appear on more than one team's board can yield query results that don't meet your expectations. Because each team can customize the board columns and swimlanes, the values assigned to work items which appear on different boards may not be the same. The primary work around for this issue is to maintain single ownership of work items by team area path.

Browser

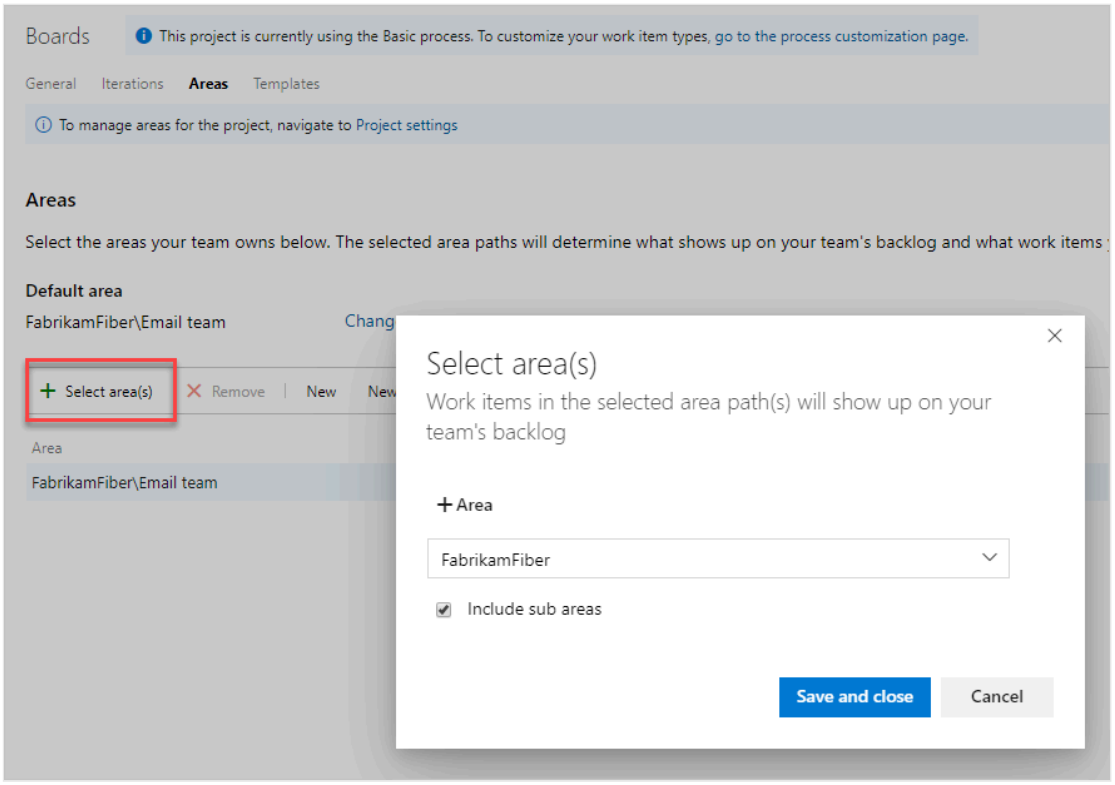
You can select existing area paths or create new or child area paths from the team settings.

1. Select **Project settings** > **Team configuration** > **Areas**.
2. Select the team you want to configure from the drop-down list at the top of the page.

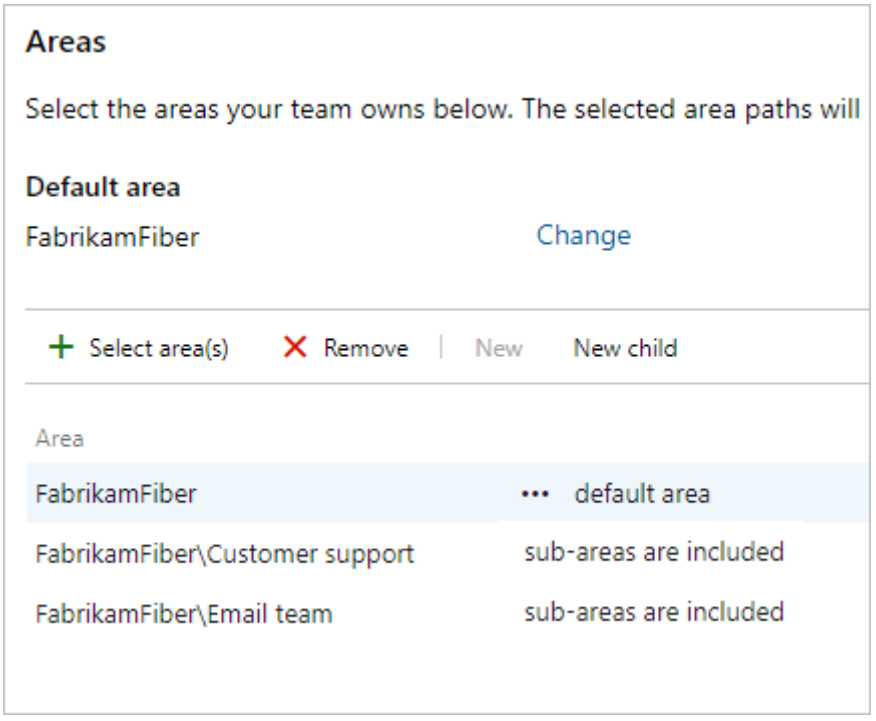
   Here, we show the Areas for the Fabrikam Fiber team.



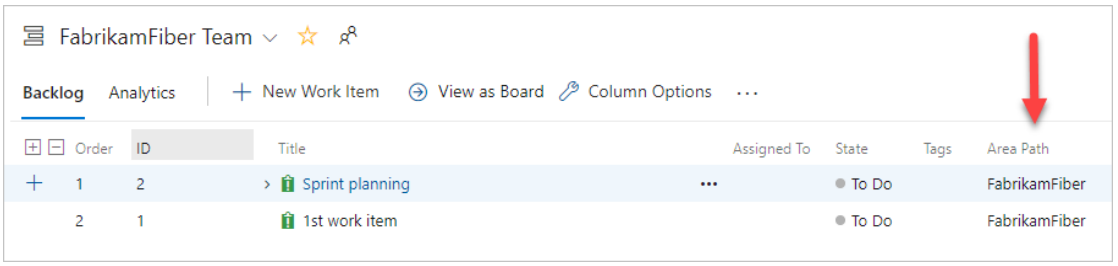
3. Choose one or more area paths to be active for the team.

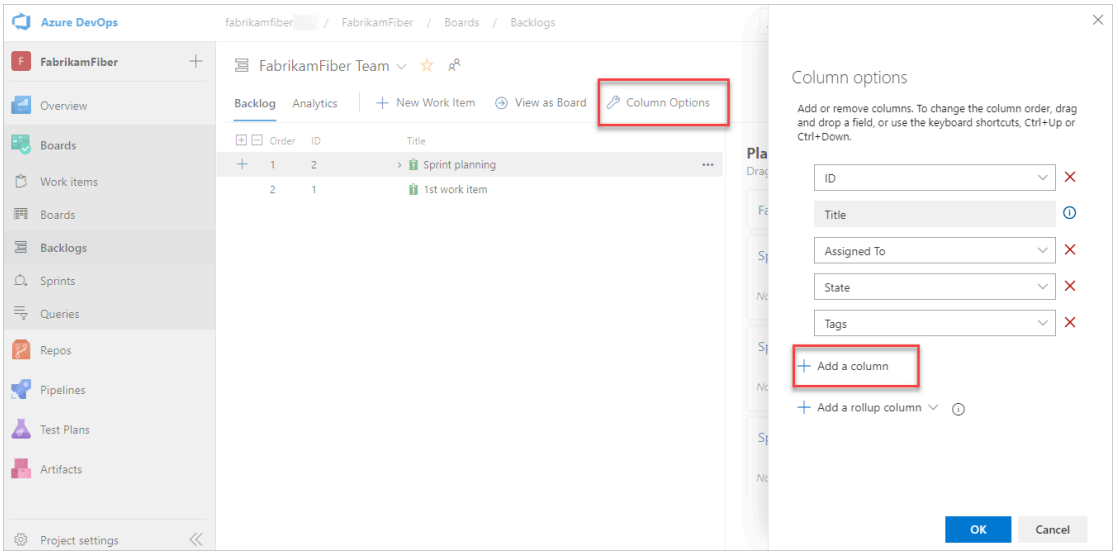


In this instance, we choose to activate the subarea paths for the project. The management team can now track progress across all teams.



4. When you finish, refresh the product backlog page for the team, and you see those work items assigned to the team. Add area path to the columns shown to see the assignments made to work items.





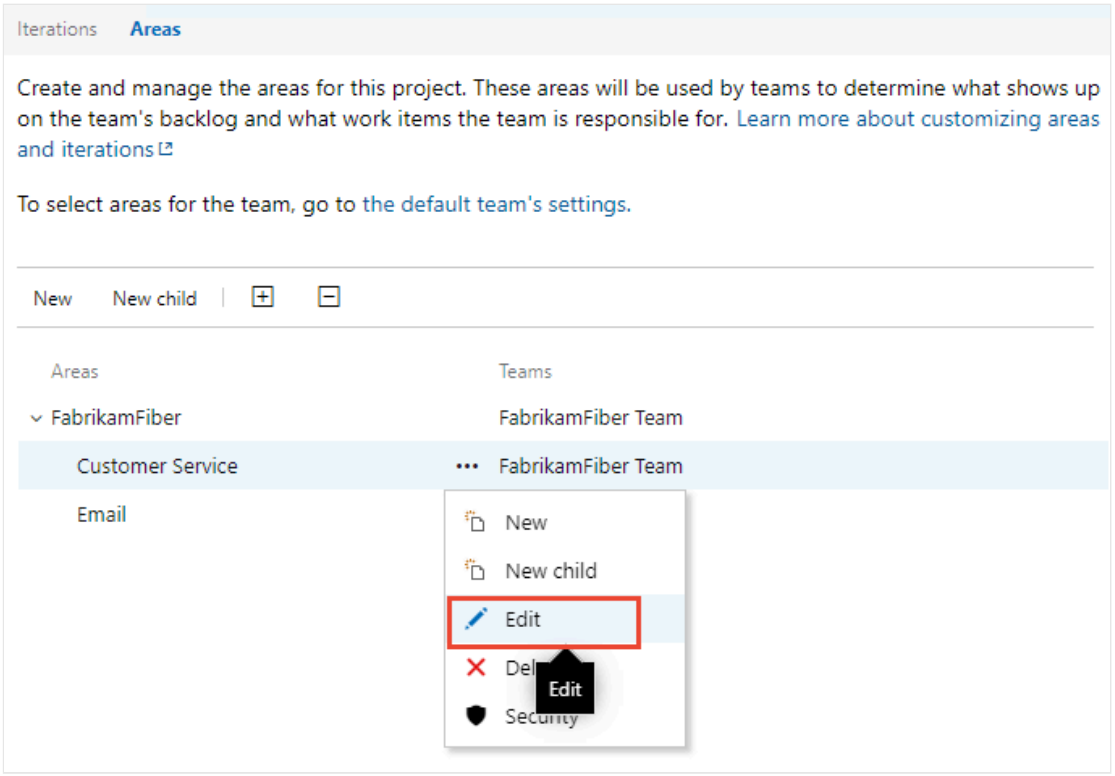
Rename or move an Area Path

The system automatically updates work items and queries that reference your updated area path. Each **Area Path** is associated with a unique GUID.

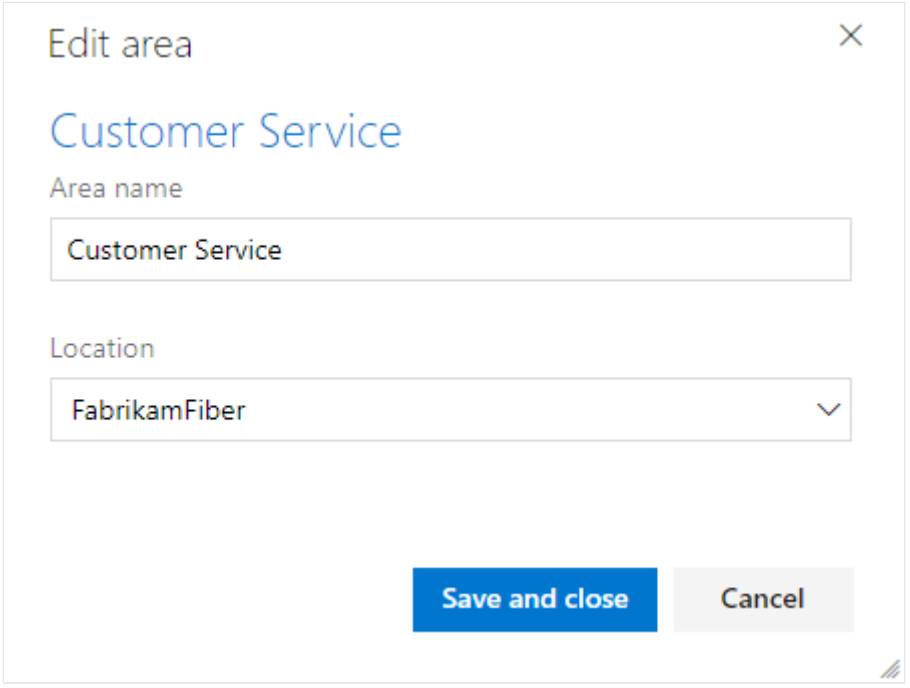
Browser

1. To rename an **Area Path**, open the web portal **Project settings>Project configuration>Areas** page.

2. Choose the ··· context menu for the node, and select **Edit**.



3. In the dialog that opens, enter the new name.



4. To move the node within the hierarchy, change the **Location** field.

> ⓘ **Note**
>
> When you rename or change the **Location** field for a node, the system automatically updates the existing work items that reference the **Area Path**.

Delete an Area Path

When you delete an **Area Path**, you must provide an **Area Path** to use to update the work items assigned to the **Area Path** you want to delete.

> ⓘ **Important**
>
> - Deleting **Area Paths** or reconfiguring **Iteration Paths** causes irreversible data loss. For example, burndown and burnup widget charts, sprint burndown, and velocity charts for teams with changed **Area Paths** won't display accurate data. Historical trend charts reference the **Area Path** and **Iteration Path** as they were defined at the time for each work item. Once deleted, you can't retrieve historical data for these paths.
> - You can only delete area and iteration paths that are no longer used by any work items.

1. To delete an **Area Path**, open the web portal **Project settings>Project configuration>Areas** page.
2. Choose the ··· context menu for the node, and select **Delete** or **Remove**.
3. In the dialog that opens, select the **Area Path** to reassign work items to, and then choose **Delete path**.

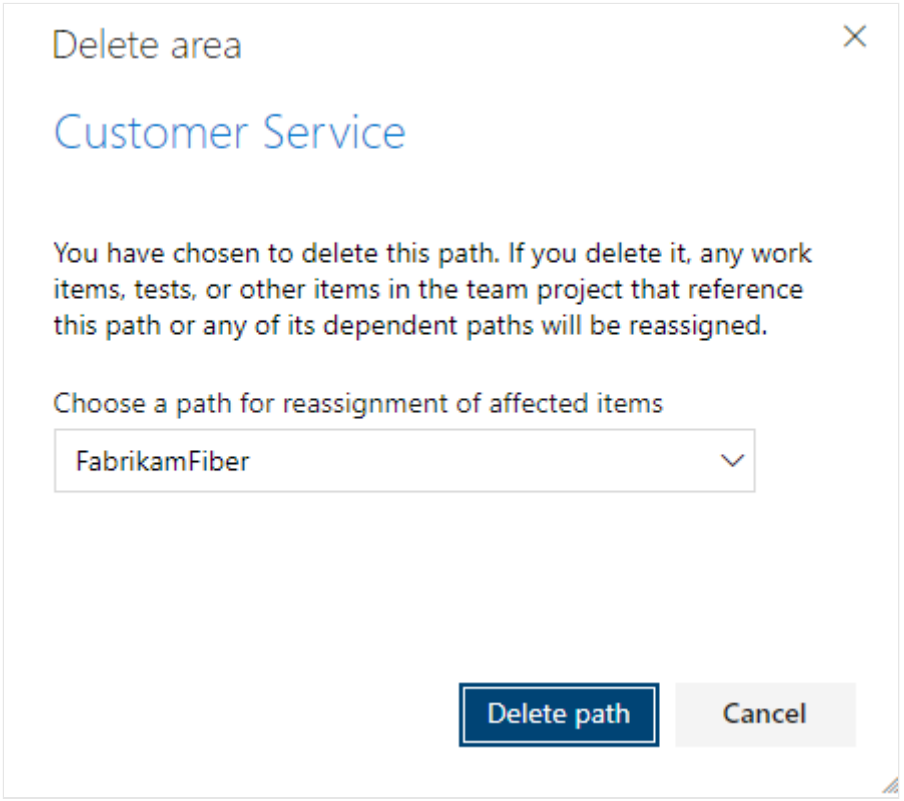
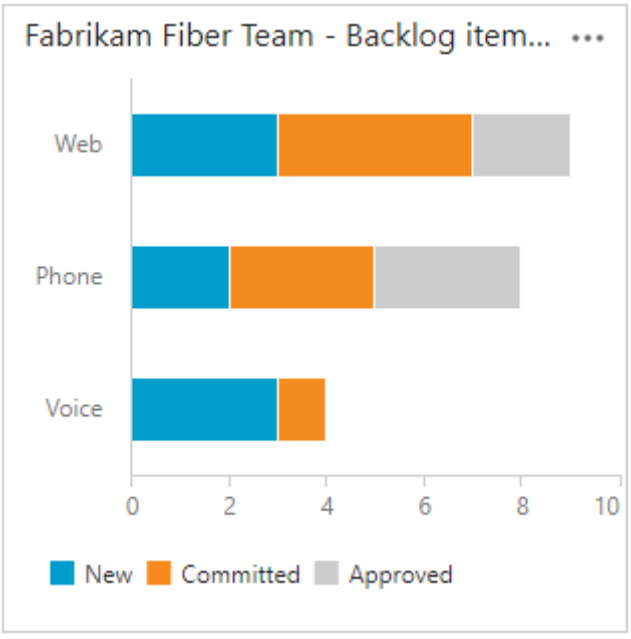


Chart progress by area

You can quickly generate queries to view the progress based on an area path. For example, visualize progress of work items that are assigned to each team's area path, as shown in the following stacked bar chart. Choose Node Name to get the leaf node of the area path.



Next steps

Set iteration paths or sprints

Related articles

As you can see, area paths play a major role in supporting Agile tools, teams, and managing work items. Learn more about working with these fields from the following articles:

- Learn about areas and iterations
- Add another team
- Configure team settings and add team administrators
- Use Agile tools that rely on areas or iterations
- Query by area or iteration path
- Set permissions and access for work tracking

Programmatic resources

Area paths and iteration paths are also referred to as *Classification Nodes*.

- Use the az boards area (Azure DevOps CLI).
- Use the Teams (REST API) to manage team configurations.
- Utilize the Classification Nodes (REST API) to organize work item classifications.

Feedback

Was this page helpful? Yes No

Provide product feedback

Define iteration paths (sprints) and configure team iterations

Article • 11/13/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Iteration Paths, also known as *sprints*, assign work items to specific time-box intervals. Define iteration paths at the project level and have each team select the paths they use. Iteration paths are shared among all selected teams. You can create a flat or hierarchical structure of iteration paths to support releases, subreleases, and sprints. If your team doesn't use sprints for planning and tracking work, retain the default team assignments and utilize product and portfolio backlogs and boards, though sprint planning tools won't be applicable.

Define iteration paths and assign them to teams when you want to use the following tools:

- Assign work items to sprints using the Planning pane
- Query and chart work items based on Iteration Path
- Forecast your product backlog
- Sprints> all tools
- Delivery Plans, a calendar view of team deliverables
- Velocity chart and Sprint burndown chart

> **Tip**
>
> If a team hasn't subscribed or selected an iteration path, that iteration path won't appear in a team view or tool.

For information about naming restrictions and limits placed on addition of Iteration Paths, see About areas and iterations, Naming restrictions.

> **Tip**
>
> To quickly update iteration dates, see **Change sprint dates**. However, if you need to define the iteration paths and tree structure, then follow the guidance provided in this article.

Prerequisites

Permissions:

- To add an iteration path to a project, be a member of the Project Administrators group. By default, the user who created the project has these permissions set.
- To add, edit, and manage iteration paths under a node, have one or more of the following permissions set to **Allow** for the node that you want to manage:
  - **Create child nodes**
  - **Delete this node**
  - **Edit this node**
  - **View permissions for this node**
- To set team iteration paths, be a team administrator or member of the Project Administrators group.

For more information about acquiring permissions, see Change project-level permissions or Set permissions and access for work tracking.

Get started

New projects start with a single root area path matching the project name. Add other area paths under this root and use predefined iteration paths for tracking work by setting dates.

If you're new to managing projects and teams, follow these steps to configure your project and teams:

1. **Configure Azure Boards**: Review Configure and customize Azure Boards.
2. **Define area paths and teams**: Follow Define area paths and assign to a team, ensuring at least one area path per team.
3. **Set iteration length**: Decide on iteration lengths, preferably consistent across teams. See About areas and iterations.
4. **Choose sprint structure**: Choose between a flat or hierarchical sprint and release structure.
5. **Define iteration paths**.
6. **Create teams**: Create necessary teams based on step 2. See Add a team, move from one default team to several teams.
7. **Assign area paths to teams**: In team configuration, assign default and other area paths to each team. Follow Set team default iteration paths.
8. **Assign iteration paths to work items**: Use bulk modify to assign work items to iteration paths for visibility on backlogs and boards. Also, see Assign backlog items to a sprint.

As needed, you can do the following tasks at any time:

- Add more child iteration nodes
- Rename an iteration path (except the root path)
- Move a child iteration path under another node
- Delete a child iteration path
- Change the default and selected iteration paths assigned to a team

Team backlog iteration versus default iteration

Each team has access to various Agile tools, as detailed in About teams and Agile tools. These tools reference the team's default area paths and selected iteration paths or sprints. Typically, teams use one area path and multiple iteration paths for work tracking, but you can use multiple area paths on backlogs and boards to support different scenarios.

Teams can set a default iteration separate from the backlog iteration. The backlog iteration determines which items appear on the team's backlogs and boards, while the default iteration assigns values to newly created work items.

All work items created within the team context are automatically assigned the team's default area path and default iteration path.

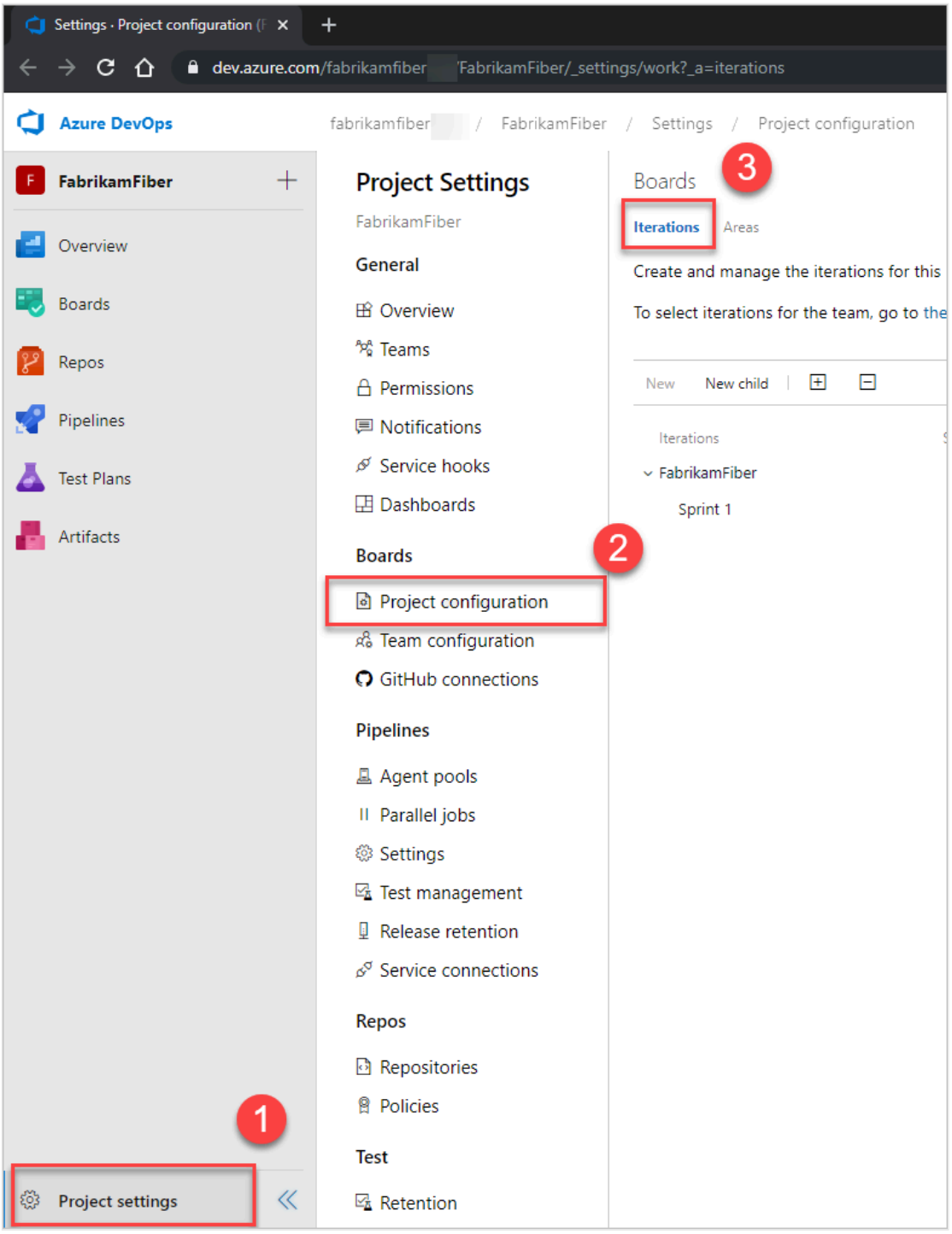
List project iterations

To list project iterations, do the following steps:

Browser

1. Sign in to your project `https://dev.azure.com/{Your_Organization/Your_Project}`.

2. Select (1) **Project settings** > (2) **Project configuration** > (3) **Iterations**.



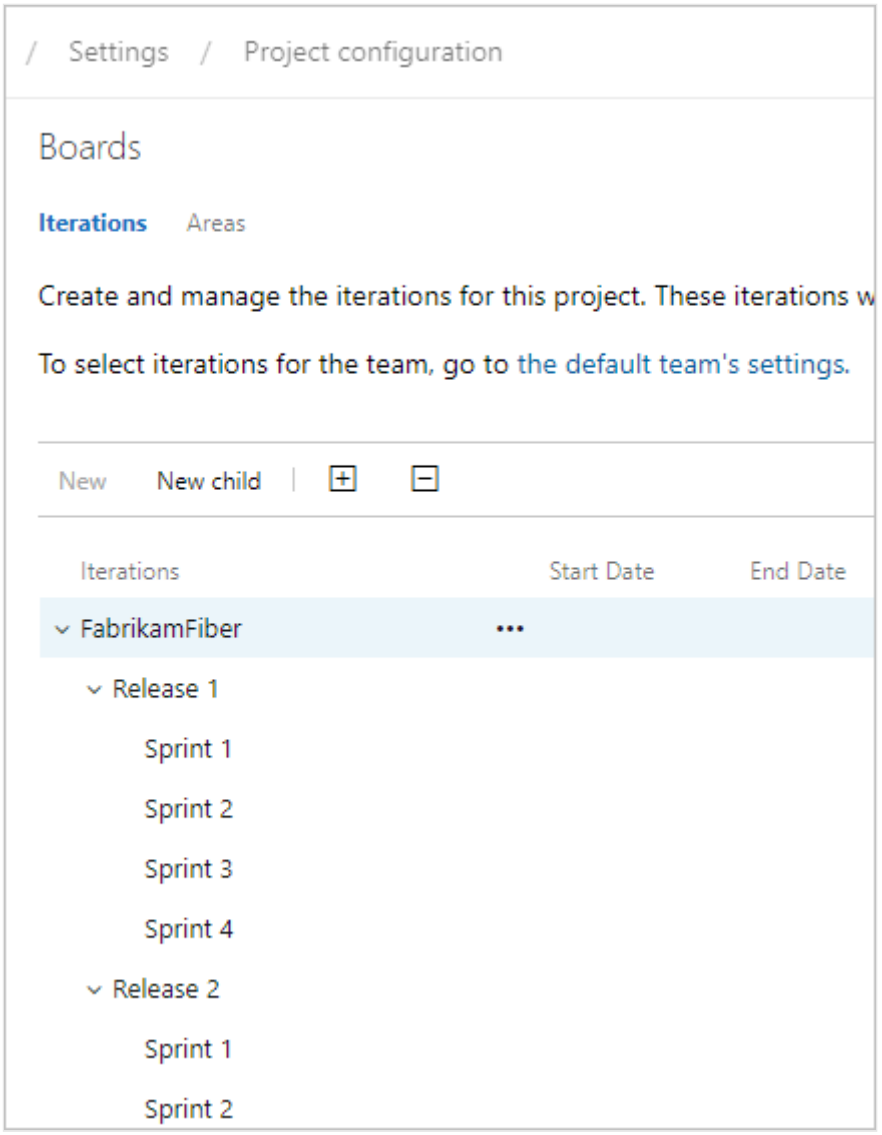
Add iterations and set iteration dates

Add iterations in the same way that you add areas. For more information about working within a sprint cadence, see Scrum and sprint planning tools.

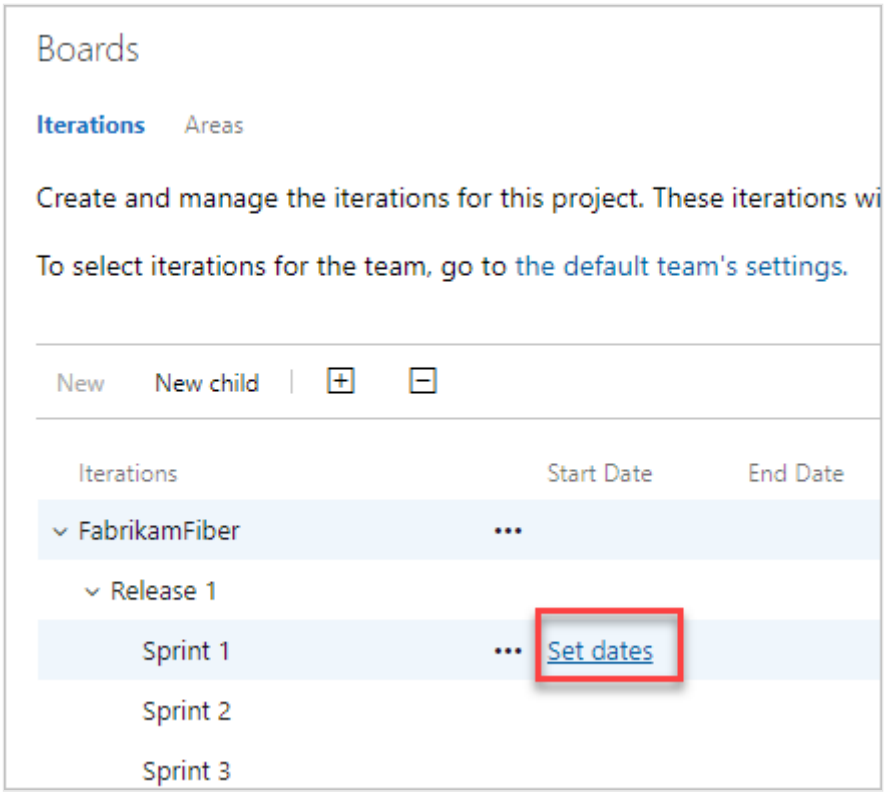
Browser

1. Add and modify iterations from **Project settings** > **Project configuration** > **Iterations**.

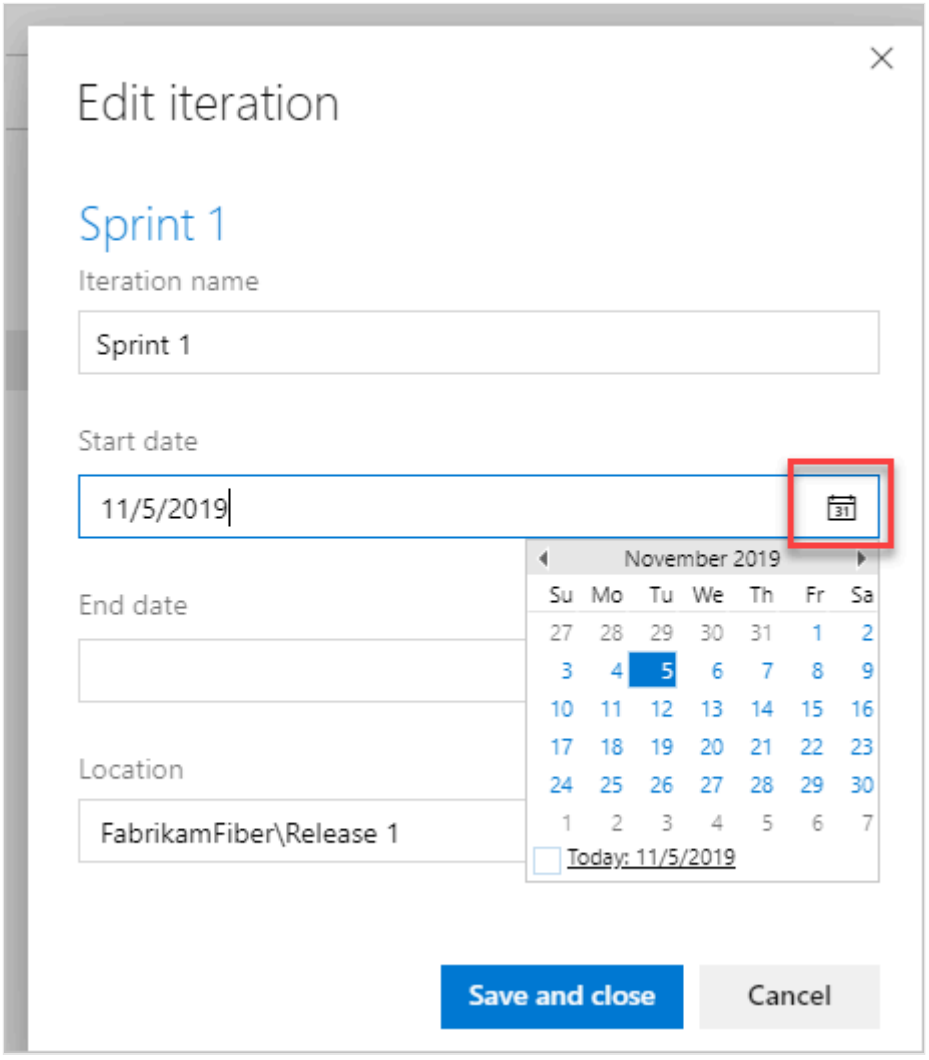
For Scrum-based projects, you see the following set of sprints.



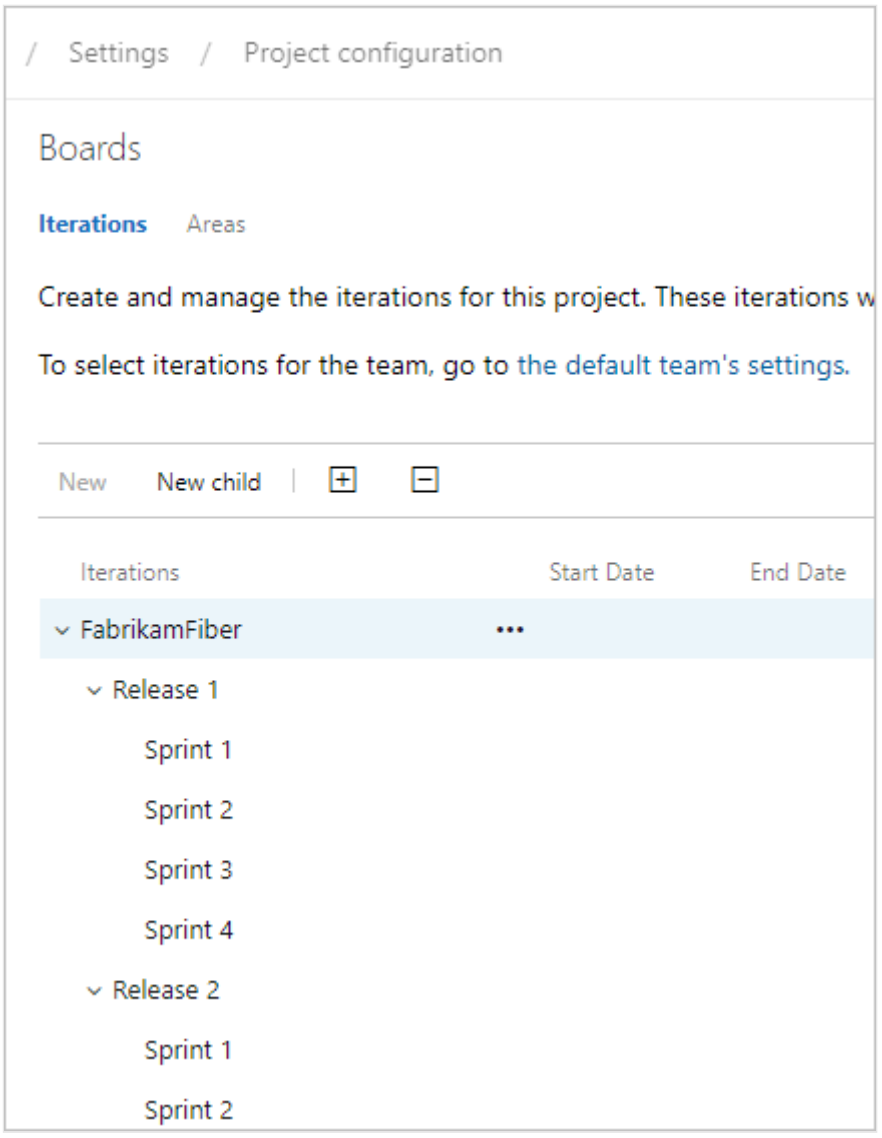
2. To schedule the start and end dates for each sprint that your teams use, highlight the sprint and choose **Set dates**. Or, select **Actions** ··· for the iteration path and choose **Edit**.



3. To select new dates, choose the calendar icon.



4. When you finish, you have a set of sprints scheduled - similar to the following image:



Your next step is to choose the sprints each team uses.

List team iterations

Set team defaults from team settings. If you're not a team administrator, get added as one. Only team or project administrators can change team settings.

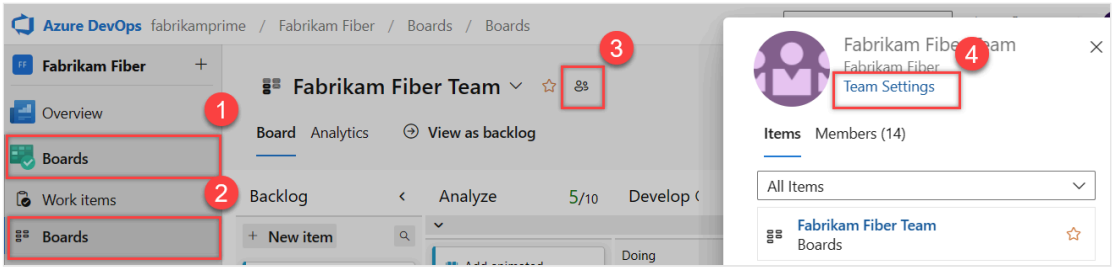
> **Note**
>
> Teams can be assigned a maximum of 300 **Iteration Paths**. For more information, see Work tracking, process, and project limits.

Browser

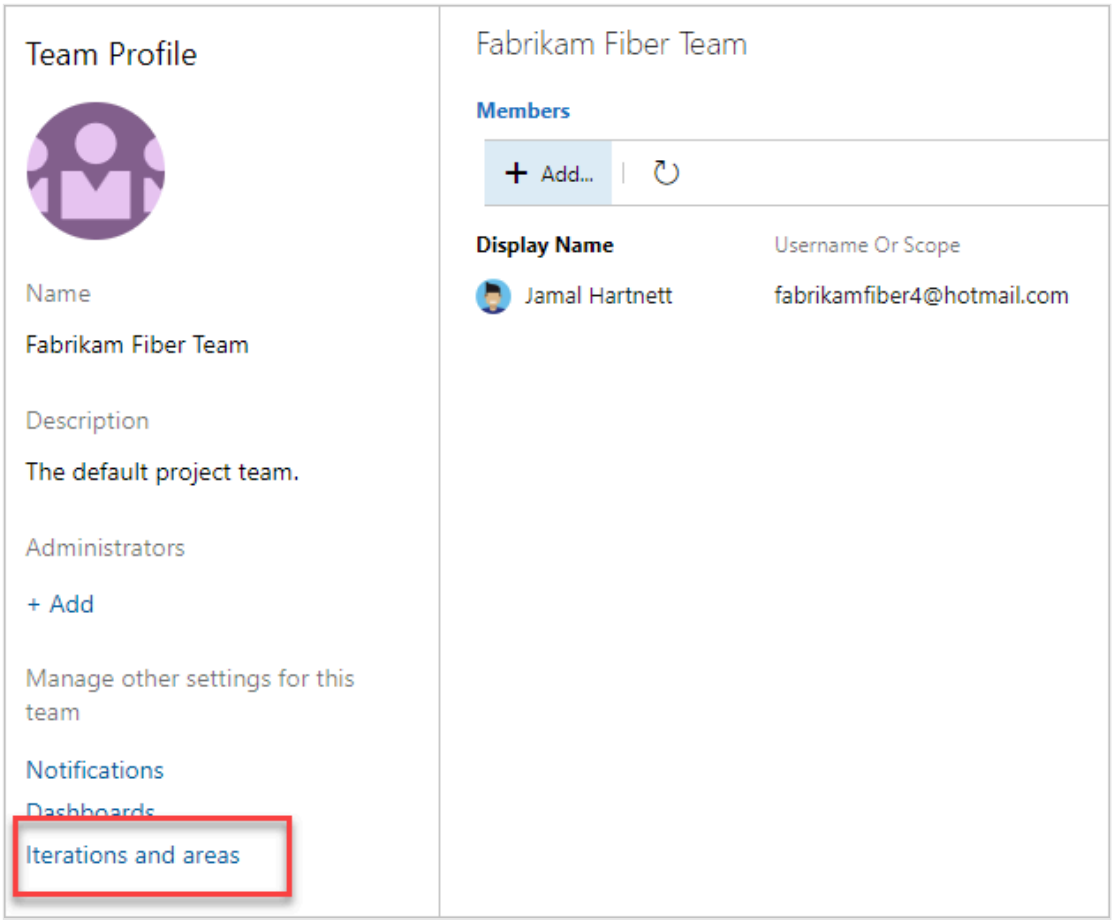
Define both areas and iterations from **Project settings** > **Boards** > **Team configuration**. Quickly navigate to it from a team work tracking backlog, board, or dashboard.

1. Open a backlog or board for a team and choose **Team profile** and then **Team Settings**.

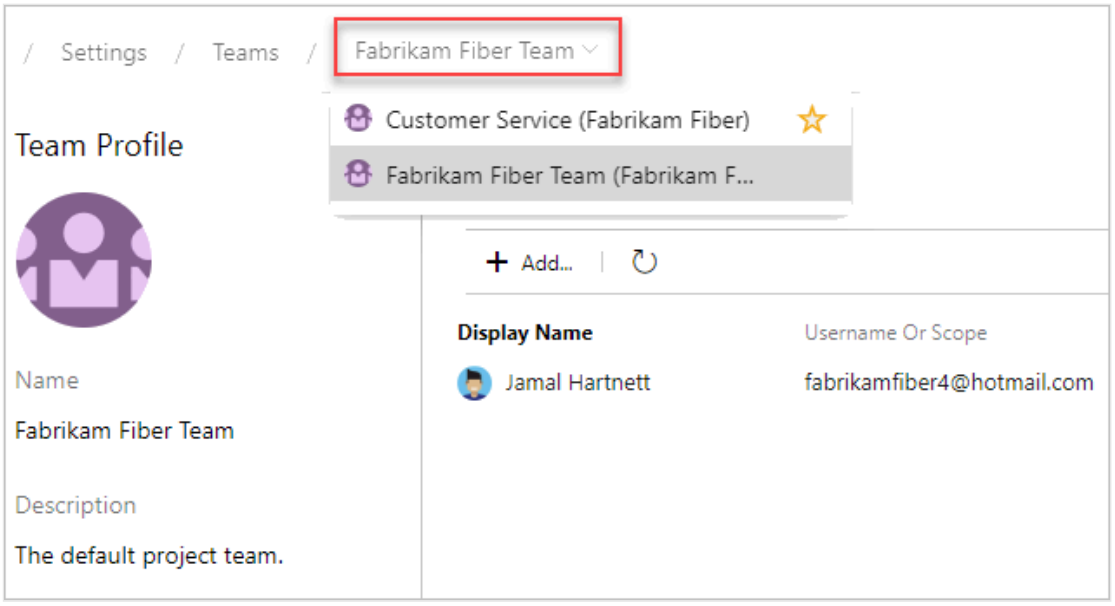
   Here we open the Board for the Web team and from there the team profile.



2. Choose **Iterations and areas**.



3. To switch the team context, use the team selector within the breadcrumbs.



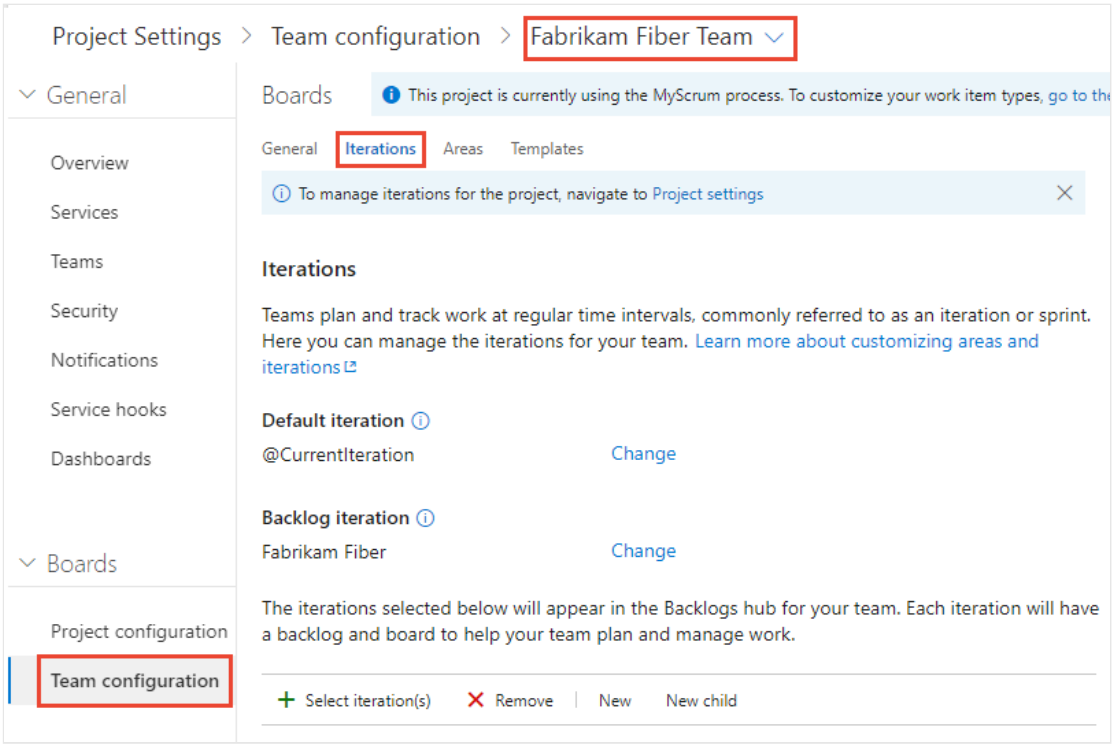
Select team sprints and set the default iteration path

You define sprints for the project and then select them to be active for each team. You assign the default iteration to use when creating new work items.

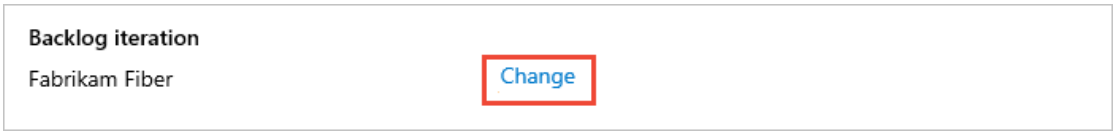
Browser

1. Open **Project settings** > **Boards** > **Team Configuration** > **Iterations** for a team.

   Here, we navigate to the Fabrikam Fiber Team.



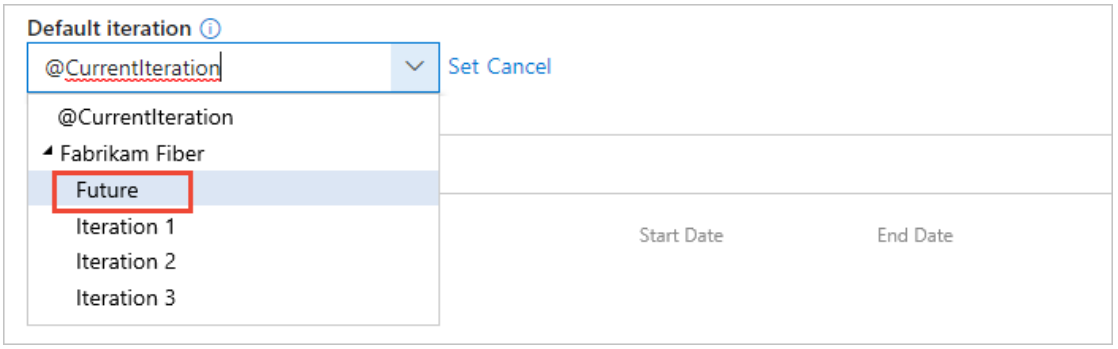
2. **Backlog iteration**. Only work items assigned to an iteration equal to or under this backlog iteration appear in the team's backlogs and boards.



   Also, all work items added through a team's backlog or board are assigned the backlog iteration.

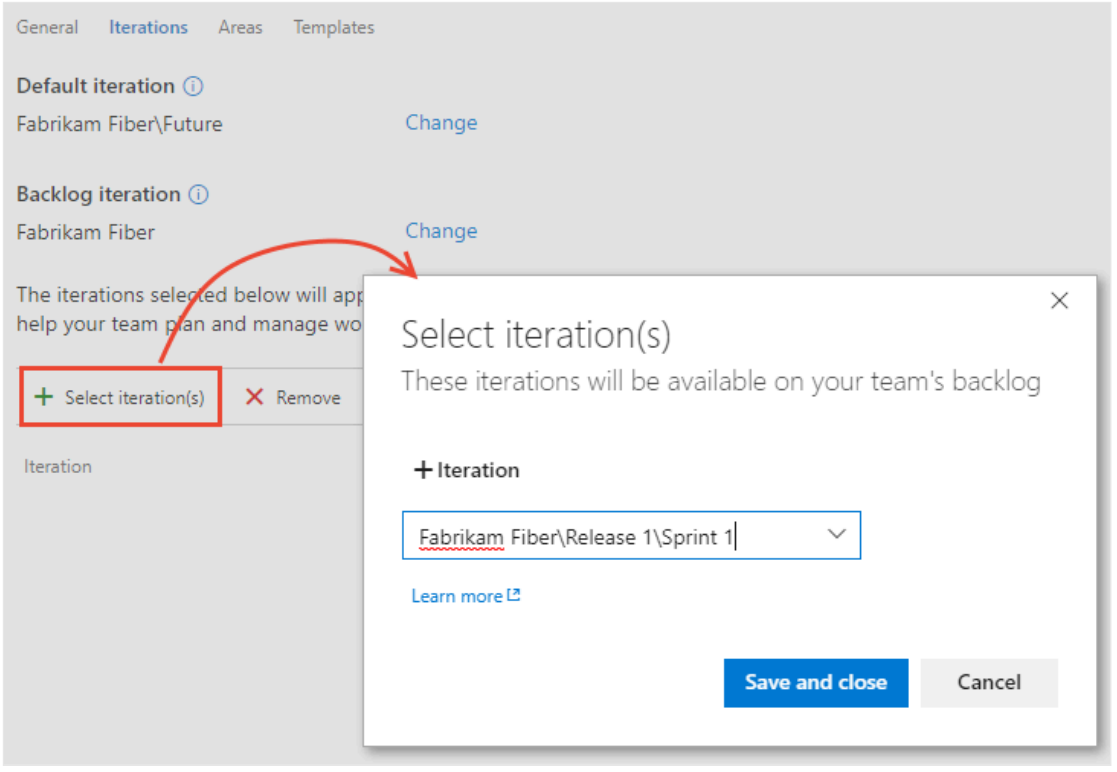
3. **Default iteration**. The default iteration defines the iteration to use when you create a work item from the team backlog or board. specify any iteration defined under the **Backlog iteration** path. To assign new work items to the current iteration, specify **@CurrentIteration**. The same macro used in queries to list work items assigned to the currently active iteration assigned to the team is used.

   For example, you might want all new work items added to a future iteration path, which you use to triage and assign to specific sprints at periodic intervals.



> **Note**
>
> New work items added through the **Work Items** page or the **New Work Items** widget on a team dashboard don't reference the **Default Iteration Path** assigned to the team. Instead, new work items are assigned the last **Iteration Path** selected by the user. New work items added through a team's **Sprints** backlog or taskboard are always assigned the **Iteration Path** associated with the selected sprint.

4. **Active sprints**. Add an iteration for each sprint backlog you want active for the team. Add each sprint, one by one, by selecting it from the menu.



When you finish, you should see a list of sprints, similar to the following.



If you don't see the sprints or dates that you need, add or edit iterations for the project, provided you have the required permissions. For more information, see Define iteration (sprint) paths.

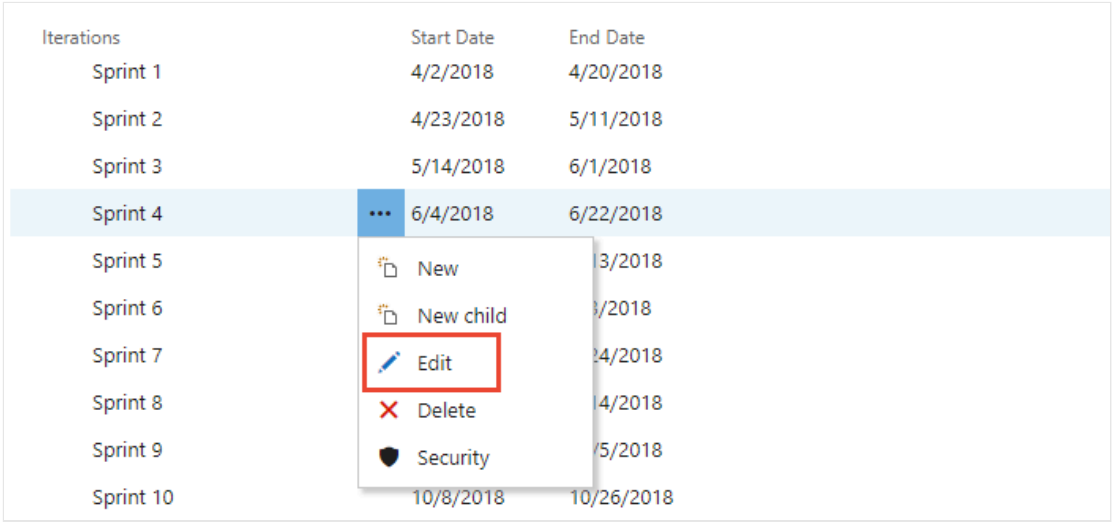
5. To see the newly activated sprint backlogs, refresh your team's product backlog page.

Rename or move an iteration

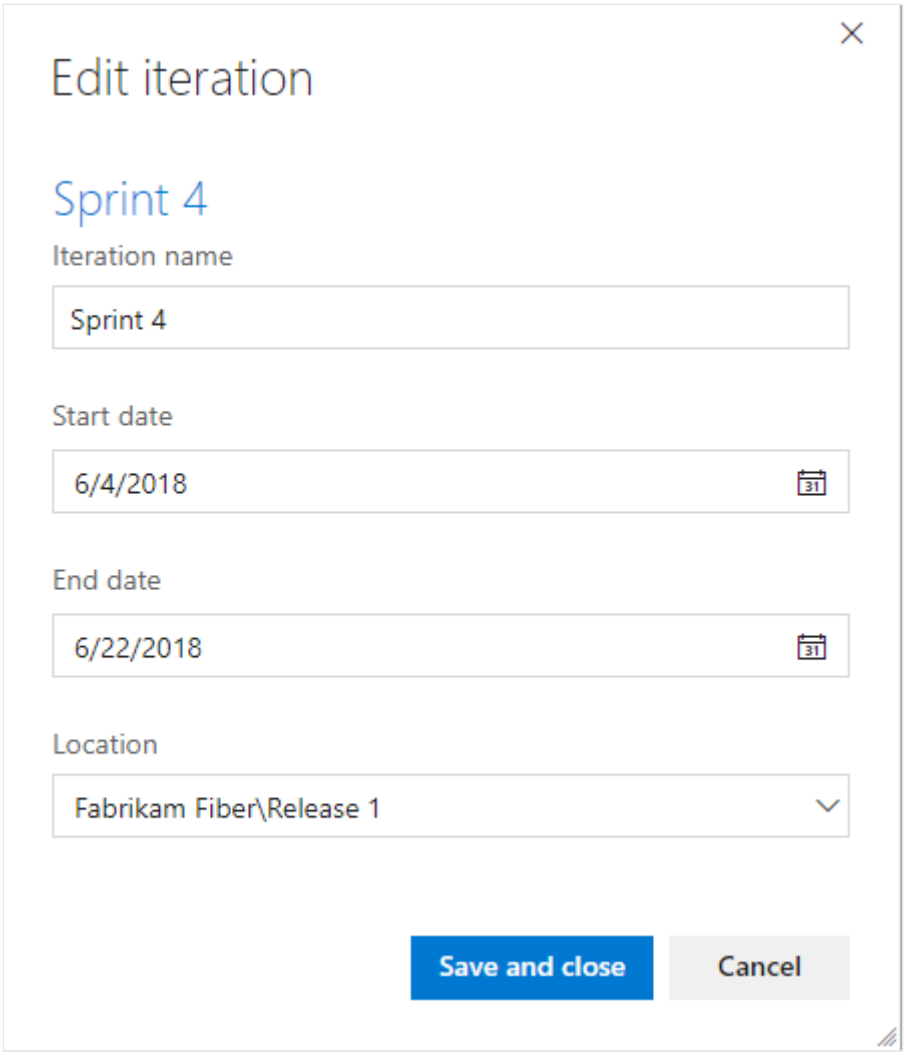
When you rename an iteration, or move the node within the tree hierarchy, the system automatically updates the work items and queries that reference the existing paths.

Browser

1. To rename an iteration path, choose the ··· context menu for the node, and then select **Edit**.



2. In the dialog that opens, enter the new name.



3. To move the node within the hierarchy, change the Location field.

4. To delete a node, choose the **Delete** option from the actions menu.

> **Note**
>
> When you delete an iteration node, the system automatically updates the existing work items with the node that you enter at the deletion prompt.

Delete an Iteration Path

> **Important**
>
> - Deleting **Area Paths** or reconfiguring **Iteration Paths** causes irreversible data loss. For example, burndown and burnup widget charts, sprint burndown, and velocity charts for teams with changed **Area Paths** won't display accurate data. Historical trend charts reference the **Area Path** and **Iteration Path** as

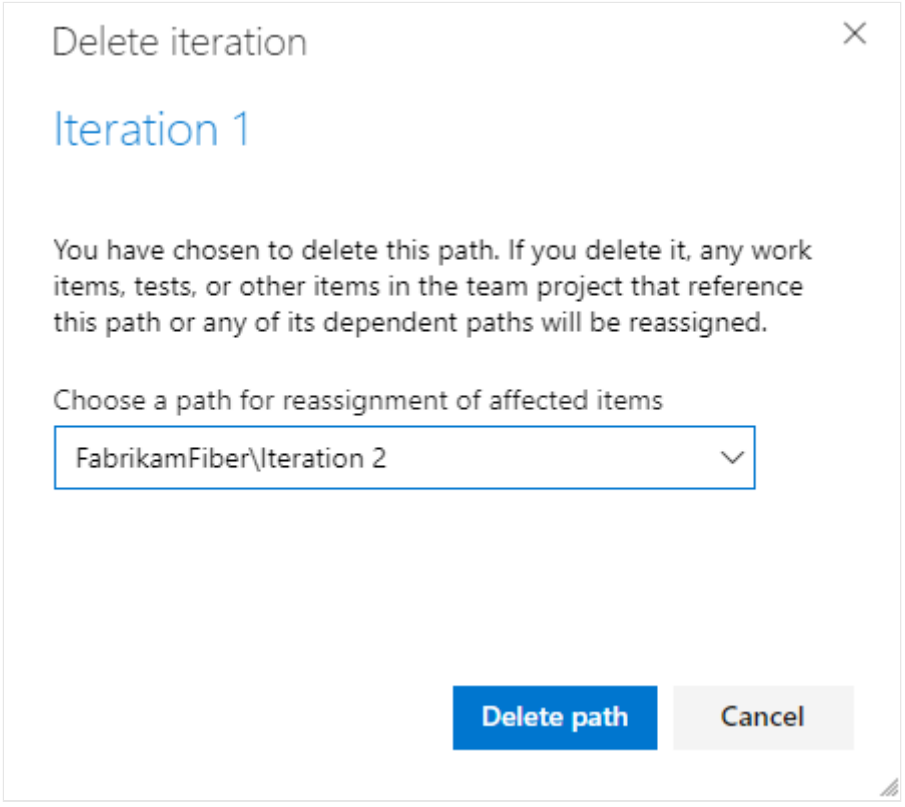
they were defined at the time for each work item. Once deleted, you can't retrieve historical data for these paths.

- You can only delete area and iteration paths that are no longer used by any work items.

Browser

When you delete an **Iteration Path**, you must provide an **Iteration Path** to use to update the work items assigned to the **Iteration Path** you want to delete.

1. To delete an **Iteration Path**, open the web portal **Project settings>Project configuration>Iterations** page.
2. Choose the ··· context menu for the node, and select **Delete** or **Remove**.
3. In the dialog that opens, select the **Iteration Path** to reassign work items to, and then choose **Delete path**.



Export/import iteration paths

You can't export or import the structure of tree paths for one project to use with another project.

Using the Azure CLI, you can use the `az boards iteration` command to list the iterations defined for one project and then add them to another project. Once you add the iterations to another project, you can then use the `az boards iteration team` command to add them to a team and set the default and backlog iteration path for the team.

You can use the Classification Nodes (REST API) and Teams (REST API) to perform similar actions.

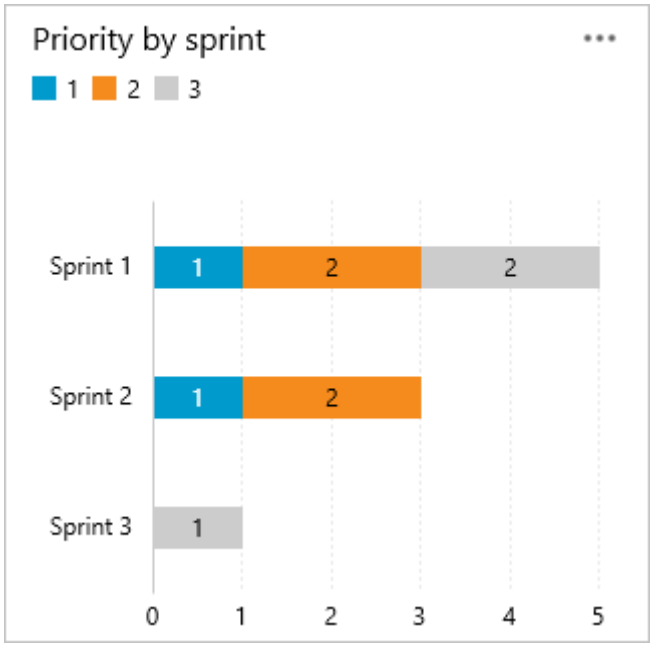
Archive iteration paths

After a while, you might want to archive iteration paths that were used for sprints that are a year or more out of date. You can do so by moving the iteration path under a node that you label "Archive". All work items are updated with the moved iteration path. Also, teams can de-select those sprints that have past. All data is maintained in the data store with the new iteration path assignments.

Before you archive the iterations, consider if you captured all the reports that you want.

Chart progress by iteration

You can quickly generate queries to view the progress for those areas. For example, you can visualize progress of work items that are assigned to sprints, as shown in the following stacked bar chart.



Related articles

- About areas and iterations
- Add another team
- Configure team settings and add team administrators
- Assign backlog items to a sprint
- Agile tools that rely on areas or iterations

Programmatic resources

Area paths and iteration paths are also referred to as *Classification Nodes*.

- Use the az boards iteration (Azure DevOps CLI)
- Use the Teams (REST API)
- Utilize the Classification Nodes (REST API)

Feedback

Was this page helpful? Yes No

Provide product feedback

Default processes and process templates

Article • 10/14/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Azure Boards offers various processes for managing work items. Selecting the right process helps optimize project workflow and ensure your project's success. In this article, explore the different processes available with Azure Boards. This article provides guidance on how to choose the most suitable process for your project.

When you create a project, you choose a *process* or *process template* based on the *process model* for which your organization or collection was created. Before you choose a process for your project, you should understand the following terms.

Expand table

Term	Description
Process model	Refers to the model used to support projects created for an organization or project collection. Only one process model is supported for a project at a time.
Process	Defines the building blocks of the work item tracking system and supports the *Inheritance* process model for Azure Boards. This model supports customization of projects through a What You See Is What You Get (WYSIWYG) user interface.
Process template	Defines the building blocks of the work item tracking system and other subsystems you access through Azure DevOps. Process templates are only used with the *Hosted XML* and *On-premises XML* process models. You can customize projects by modifying and importing process template XML definition files.

The default process types are *Basic*, *Agile*, *Capability Maturity Model Integration (CMMI)*, and *Scrum*. The work tracking objects in the default processes and process templates are the same. They're summarized in this article.

Default processes

The default processes differ mainly in the work item types they provide for planning and tracking work. The default processes are:

- **Basic**: Is the most lightweight and is in a selective preview.
- **Scrum**: Is the next most lightweight.
- **Agile**: Supports many Agile method terms.

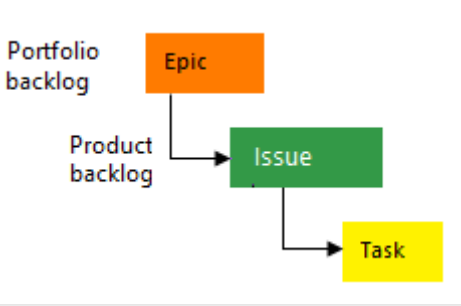
- **CMMI**: Provides the most support for formal processes and change management.

> **Note**
>
> The Basic process is available with Azure DevOps Server 2019 Update 1 and later versions.

Basic

Choose Basic when your team wants the simplest model that uses Issue, Task, and Epic work item types to track work.

Tasks support tracking Remaining Work.

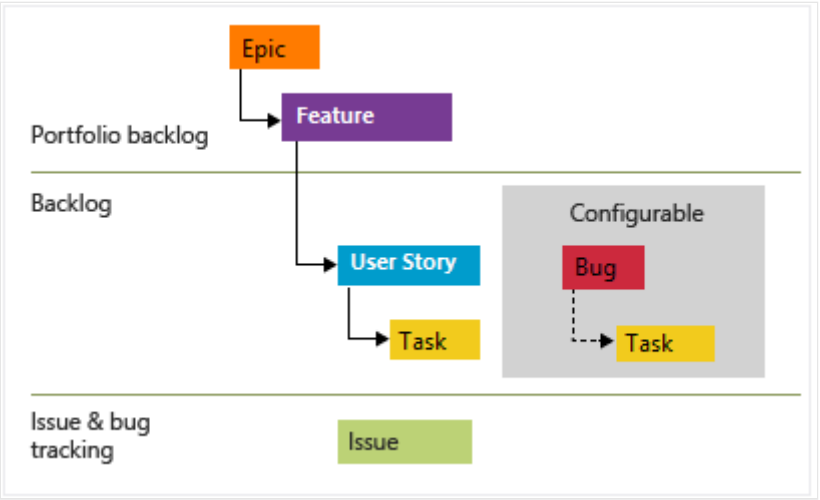


Agile

Choose Agile when your team uses Agile planning methods, including Scrum, and tracks development and test activities separately. This process works great for tracking User Stories and, optionally, bugs on the board. You can also track bugs and tasks on the taskboard.

For more information about Agile methodologies, see Agile Alliance.

Tasks support tracking Original Estimate, Remaining Work, and Completed Work.

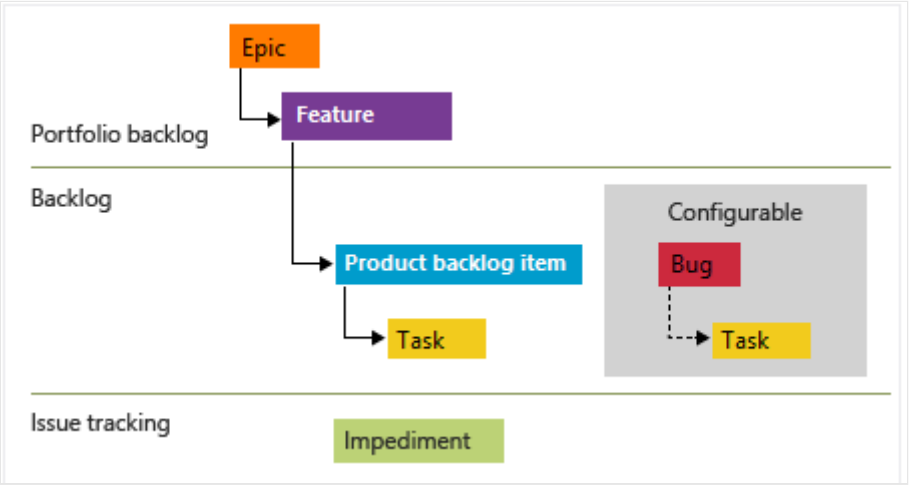


Scrum

Choose Scrum when your team practices Scrum. This process works great for tracking product backlog items and bugs on the board. You can also break down product backlog items and bugs into tasks on the taskboard.

This process supports the Scrum methodology as defined by the Scrum organization.

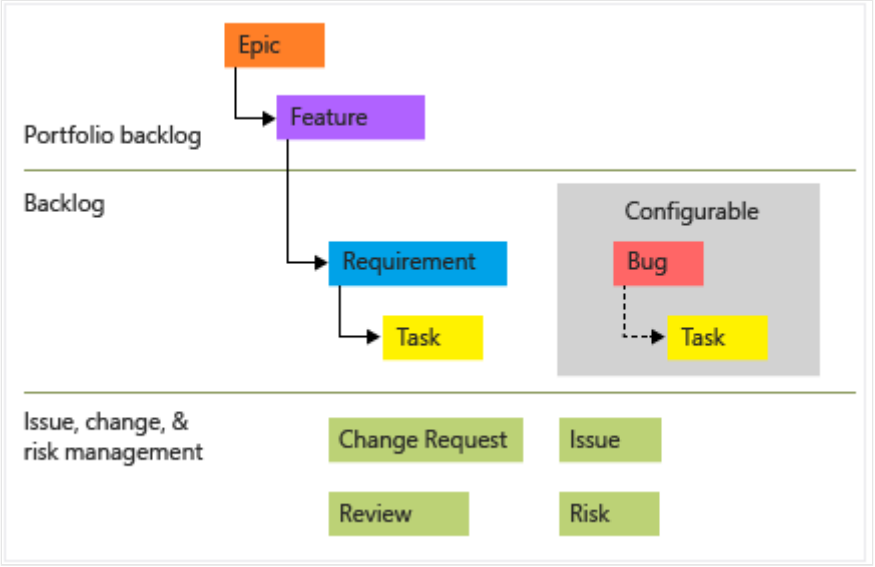
Tasks support tracking Remaining Work only.



CMMI

Choose CMMI when your team follows more formal project methods that require a framework for process improvement and an auditable record of decisions. With this process, you can track requirements, change requests, risks, and reviews.

This process supports formal change management activities. Tasks support tracking Original Estimate, Remaining Work, and Completed Work.



If you need more than two or three backlog levels, add more based on the process model that you use:

- **Inheritance**: Customize your backlogs or boards for a process
- **Hosted XML or On-premises XML**: Add portfolio backlogs

Main distinctions among the default processes

The default processes are designed to meet the needs of most teams. If your team has unusual needs and connects to an on-premises server, customize a process and then create the project. You can also create a project from a process and then customize the project.

The following table summarizes the main distinctions between the work item types and states used by the four default processes.

Tracking area

Basic

Agile

Scrum

CMMI

Workflow states

- To Do
- Doing
- Done
- New
- Active
- Resolved
- Closed
- Removed
- New
- Approved
- Committed
- Done
- Removed
- Proposed
- Active
- Resolved
- Closed

Product planning (see Note 1)

- Issue
- User Story
- Bug (optional)
- Product backlog item
- Bug (optional)
- Requirement
- Bug (optional)

Portfolio backlogs (see Note 2)

- Epic
- Epic
- Feature
- Epic
- Feature
- Epic
- Feature

Task and sprint planning (see Note 3)

- Task
- Task

- Bug (optional)
- Task
- Bug (optional)
- Task
- Bug (optional)

Bug backlog management (see Note 1)

- Issue
- Bug
- Bug
- Bug

Issue and risk management

- Issue
- Issue
- Impediment
- Issue
- Risk
- Review

Notes:

1. Add work items from the product backlog or board. The product backlog shows a single view of the current backlog of work that can be dynamically reordered and grouped. Product owners can prioritize work and outline dependencies and relationships. Each team can configure how they want bugs to show up on their backlogs and boards.
2. Define a hierarchy of portfolio backlogs to understand the scope of work across several teams and see how that work rolls up into broader initiatives. Each team configures which portfolio backlogs appear for their use.
3. Define tasks from the sprint backlog and taskboard. With capacity planning, teams can determine if they're over capacity or under capacity for a sprint.

Workflow states, transitions, and reasons

Workflow states support tracking the status of work as it moves from a `New` state to a `Closed` or a `Done` state. Each workflow consists of a set of states, the valid transitions between the states, and the reasons for transitioning the work item to the selected state.

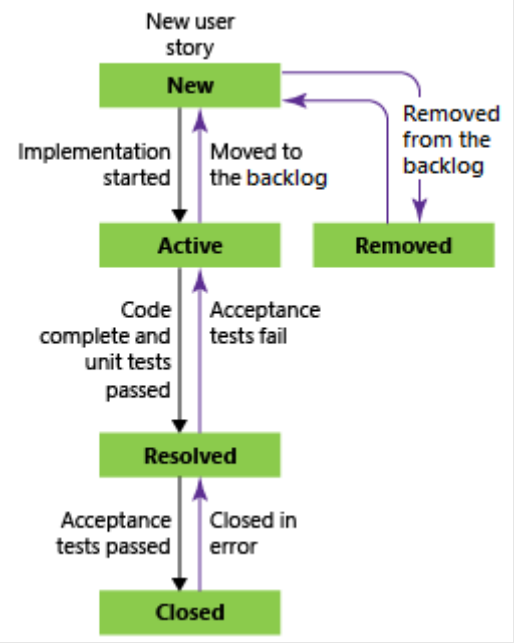
> **Important**
>
> For Azure DevOps Services and Azure DevOps Server 2019, the default workflow transitions support any state to any state transition. Customize these workflows to restrict some transitions. For more information, see Customize your work tracking experience.
>
> View the supported workflow transitions for each work item type by installing the State Model Visualization Marketplace extension. This extension adds a new hub under **Boards** labeled **State Visualizer**. On that page, choose a work item type and view the workflow state model.

The following diagrams show the typical forward progression of those work item types used to track work and code defects for the three default processes. They also show some of the regressions to former states and transitions to removed states.

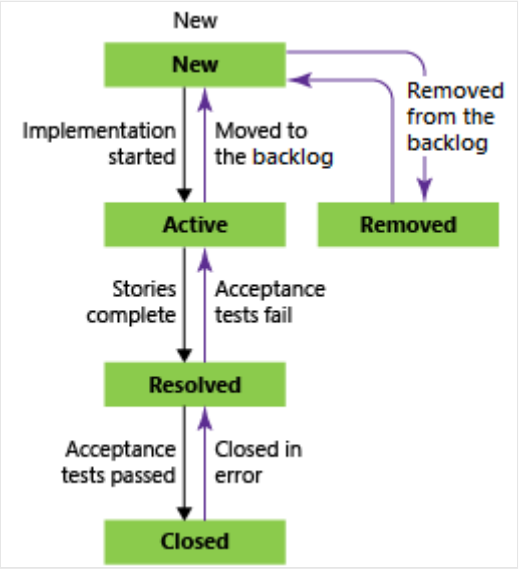
Each image shows only the default reason associated with the transition.

Agile process

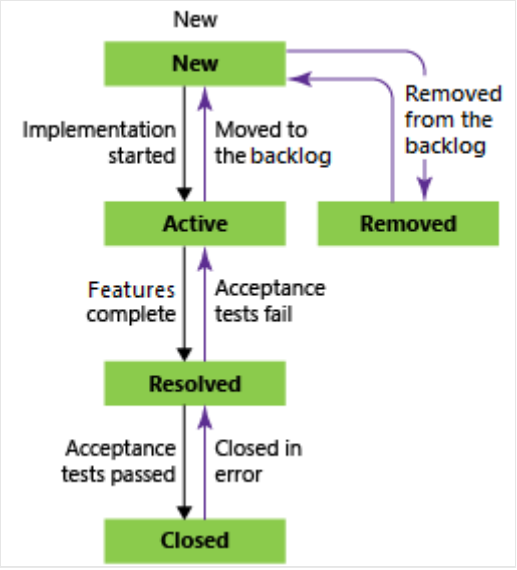
User Story



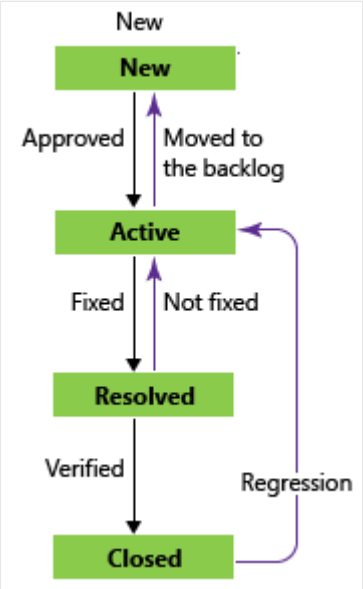
Feature


New
New
Removed from the backlog
Implementation started
Moved to the backlog
Active
Removed
Stories complete
Acceptance tests fail
Resolved
Acceptance tests passed
Closed in error
Closed

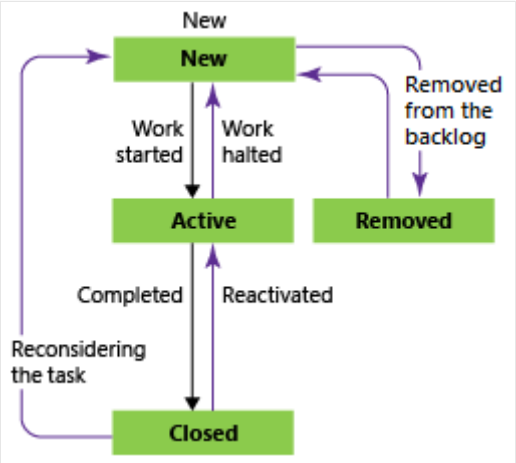
Epic


New
New
Removed from the backlog
Implementation started
Moved to the backlog
Active
Removed
Features complete
Acceptance tests fail
Resolved
Acceptance tests passed
Closed in error
Closed

Bug



Task



Most work item types used by Agile tools, the ones that appear on backlogs and boards, support any-to-any transitions. Update the status of a work item by using the board or the taskboard. Drag a work item to its corresponding state column.

Change the workflow to support other states, transitions, and reasons. For more information, see Customize your work tracking experience.

Work item states

When you change the state of a work item to `Removed`, `Closed`, or `Done`, the system responds as follows:

- `Closed` or `Done`: Work items in this state don't appear on the portfolio backlog and backlog pages. They do appear on the sprint backlog pages, board, and taskboard. When you change the portfolio backlog view to **Show backlog items**, for example,

to view features to product backlog items, work items in the `Closed` and `Done` state appear.

- `Removed`: Work items in this state don't appear on any backlog or board.

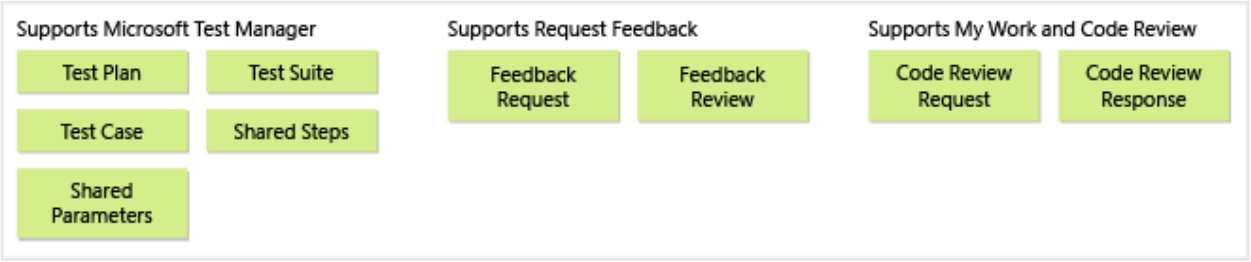
Your project maintains work items as long as the project is active. Even if you set work items to `Closed`, `Done`, or `Removed`, the data store keeps a record. Use a record to create queries or reports.

> **Note**
>
> Completed or closed work items don't display on the backlogs and boards after their **Changed Date** value is greater than 183 days (about a half a year). You can still list these items by using a query. If you want them to show up on a backlog or board, you can make a minor change to them, which resets the clock.

If you need to permanently delete work items, see Remove or delete work items.

Work item types added to all processes

The following work item types are added to all processes except the Basic process.



Your team can create and work with these types by using the corresponding tool.

Expand table

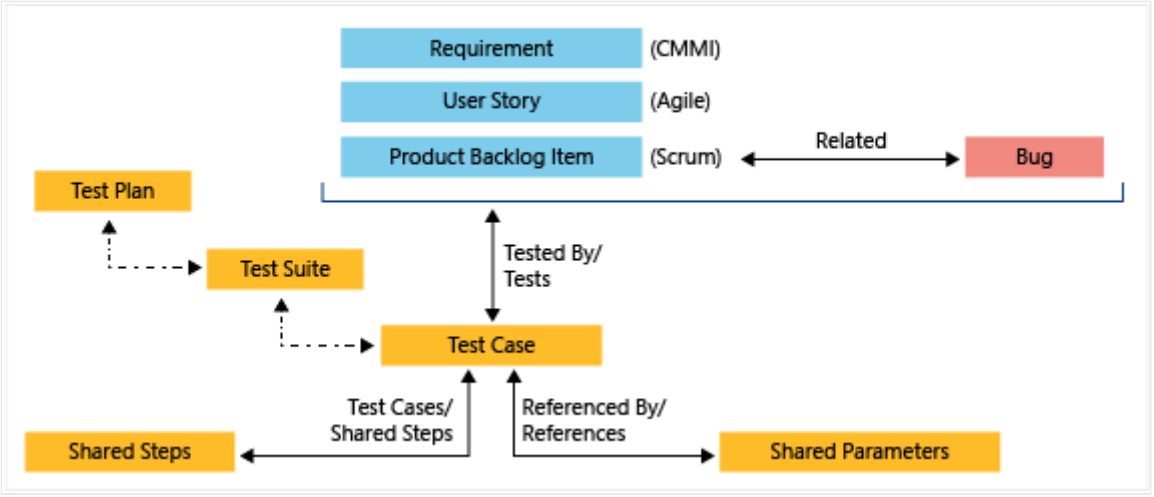
Tool	Work item types
Microsoft Test Manager	`Test Plan`, `Test Suite`, `Test Case Shared Steps`, `Shared Parameters`
Request Feedback	`Feedback Request`, `Feedback Response`
My Work (from Team Explorer), Code Review	`Code Review Request`, `Code Review Response`

Work items from these type definitions aren't meant to be created manually and are then added to the `Hidden Types` category. Work item types added to the `Hidden Types`

category don't appear on the menus that create new work items.

Work item types that support the test experience

Work item types that support the test experience and work with Test Manager and the web portal are linked together by using the link types shown in the following image.



From the web portal or Microsoft Test Manager, view which test cases are defined for a test suite and view which test suites are defined for a test plan. However, these objects aren't connected to each other through link types. Customize these work item types as you would any other work item types. For more information, see Customize your work tracking experience.

If you change the workflow for the test plan and test suite, you might need to update the process configuration as described here. For definitions of each test field, see Create a query based on build and test integration fields.

Related articles

- Customize your work tracking experience
- Upload and download process templates

Feedback

Was this page helpful? Yes No

Provide product feedback

About configuring and customizing Azure Boards

Article • 03/15/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

You can configure and customize Azure Boards in many ways, to better manage your portfolio, dependencies, and monitoring. We recommend the tasks covered in this article especially for administrators who are responsible for managing multi-team projects.

Quick access to common tasks:

Customize cards | Manage columns | Expedite work with swimlanes | Configure your backlog.

> **Note**
>
> Most of the guidance in this article is valid for both the cloud and on-premises versions. However, some of the features included in this article, such as Rollup, Analytics, and some portfolio planning tools, are only available for the cloud at this time.

If you're just getting started as a Project Administrator, see also Get started as an administrator.

Considerations

To make the most of Azure Boards, understand how your teams use their tools and functions (for example, Scrum, Kanban, and Scrumban), and their dependencies on configurations and customizations. The following table summarizes the primary items you should consider as you structure your project.

Project level

- How many teams you want to define
- How to structure area paths to support portfolio management views
- Field customizations
- Custom work item types (WITs)

- Portfolio backlog customizations
- Workflow customizations

Team level

- How you use your product backlog to plan and prioritize your work
- Whether you track bugs as requirements or as tasks, or not use bugs at all
- Whether or not you use tasks to track time and capacity
- How you use portfolio backlog levels
- How you inform upper management of progress, status, and risks

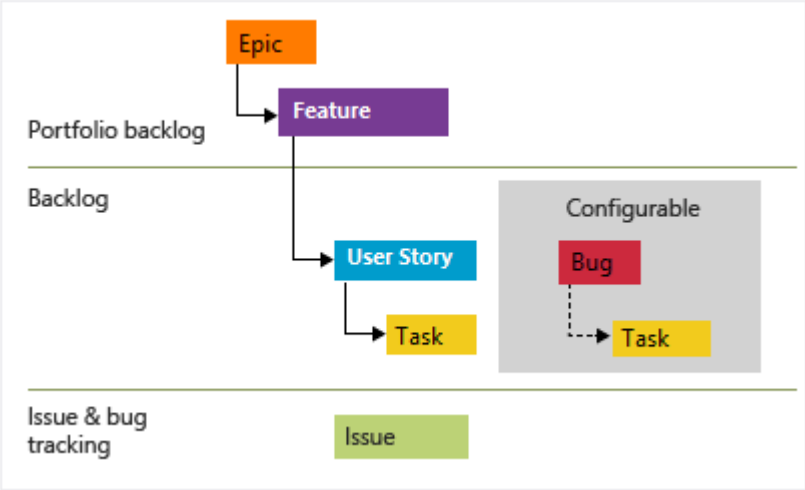
Customize your work tracking building blocks and tools to support business needs and communicate the usage guidelines to your team.

Work item types and portfolio backlogs

When you create a project in Azure Boards, you first select a process. Each process (Agile, Basic, Scrum, and CMMI) supports a hierarchy of WITs, including product and portfolio backlogs. Default WITs for each process are listed in corresponding tabs, with backlogs under **Requirements** and tasks under **Task**.

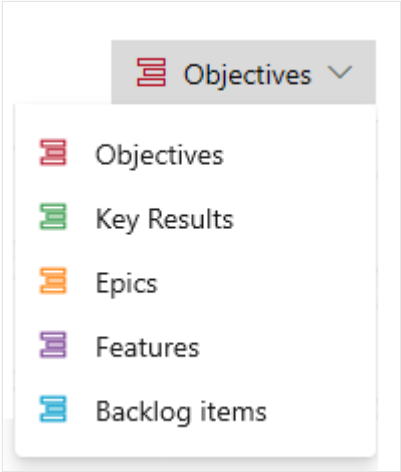
Agile process

The following image shows the Agile process backlog work item hierarchy. User Stories and Tasks are used to track work, Bugs track code defects, and Epics and Features are used to group work under larger scenarios.



Each team can configure how they manage Bugs—at the same level as User Stories or Tasks—by configuring the Working with bugs setting. For more information about using these work item types, see Agile process.

You can add custom WITs at each level, and even add custom portfolio backlogs. Here, for example, is a project that added Objectives and Key Results as custom WITs and corresponding portfolio backlogs to the Scrum process.



Work tracking options and recommended usage

Teams can choose which WITs they use to track their work. The following table summarizes the main options, recommended usage, and supported tasks and tools.

Work tracking options	Tasks and tools supported
Tasks only **Not recommended** There's no way to quickly enter new tasks in a backlog nor prioritize a backlog of tasks. Also, there's no support for calendar views, cross-team views, or portfolio planning	
Requirements with child-dependent tasks **Supports Scrum methods** Recommended for teams that follow Scrum methods and want to track time associated with work.	• Quickly define and prioritize backlog items: Product backlog • Forecast sprints using team velocity: Forecast • Plan sprints: Backlog Planning tool • Plan and track capacity: Sprint capacity tool • Track estimated and remaining work: Taskboard

- Monitor sprint burndown based on remaining work such as hours or days: Sprint burndown
- Conduct daily scrums, update, and monitor task status: Sprint Taskboard
- Estimate work: Define Story Points, Effort, or Size
- View progress bars, counts, or sums of rollup on tasks: Rollup
- Track dependencies across teams and projects: Delivery Plans

Many teams start out using Scrum methods to track and plan their work using the tools available through the Sprints hub. The Sprints tools support estimating and tracking remaining work and use of capacity planning. If you don't plan on using these tools, then adding child-dependent tasks is optional. Developers might add them as a checklist of items they need to complete a user story or backlog requirement.

Requirements only, such as user stories (Agile), issues (Basic), product backlog items (Scrum), requirements (CMMI)

Supports Kanban and Scrumban methods
Recommended for teams that follow Kanban or Scrumban methods, estimate work using Story Points, Effort, or Size, and don't track time associated with work.

- Quickly define and prioritize backlog items: Product backlog
- Plan sprints: Backlog Planning tool
- Estimate work: Define Story Points, Effort, or Size
- Forecast sprints using team velocity: Forecast
- Monitor sprint burndown based on requirement estimates: Sprint burndown
- Update requirement status: Kanban board
- Track dependencies across teams and projects: Delivery Plans

Requirements grouped under portfolio WITs, such as epics and features

Supports calendar views, cross-team views, and portfolio planning
Recommended for organizations with several teams that want to view rollups and calendar views associated with multiple teams, and take advantage of all portfolio planning tools.

- Quickly define and prioritize portfolio items: Portfolio backlogs
- Quickly define child user stories of portfolio items: Portfolio checklists
- Map work items to features and epics: Mapping tool
- View cross-team progress calendar view: Delivery plans
- View calendar view of all team features: Feature Timeline
- View calendar view of a specific epic: Epic Roadmap
- View progress bars, counts, or sums of rollup on child items: Rollup
- Track dependencies across teams and projects: Delivery Plans

Options to configure and customize

The following table shows the areas you can configure and customize and the tools impacted by those customizations. You can customize each area at the Organization, Project, or Team level as noted, or a combination of two. For a description of the Standard tools, Analytics tools, and Portfolio planning tools, see What is Azure Boards, In-context reports: Work tracking, and Plans (Agile at scale).

Configure or customize

Standard tools

Analytics

Portfolio planning tools

Area paths, project configuration, and team subscriptions (Project, Team)

- Boards>All tools
- Backlogs>All tools
- Sprints>All tools
- Cumulative flow diagram
- Velocity
- Burndown trend
- Delivery plans
- Feature timeline
- Epic Roadmap
- Portfolio plans (Beta)

Iteration paths, project configuration, and team subscription (Project, Team)

- Backlogs>Sprint planning
- Sprints>Sprint backlogs
- Sprints>Sprint capacity
- Sprints>Taskboard
- Velocity
- Burndown trend
- Delivery plans
- Feature timeline
- Epic Roadmap
- Portfolio plans (Beta)

Show bugs on backlogs & boards (Team)
Custom WITs, Product backlog (Process)
Custom WITs, Taskboard (Process)

- Boards>Product board
- Backlogs>Product backlog
- Backlogs> Mapping tool
- Sprints>Sprint backlogs
- Sprints>Taskboard
- Velocity

Custom WITs, Portfolio backlog (Process)
More portfolio backlogs (Process)

- Boards>Portfolio boards
- Backlogs>Portfolio backlogs
- Backlogs> Mapping tool
- Velocity

Custom workflow (Process)

- Boards>Product board
- Boards>Portfolio boards
- Sprints>Taskboard
- Cumulative flow diagram

Custom field (Process)

- Boards>Product board
- Boards>Portfolio boards
- Rollup progress bars, sum, or count

Area paths, product teams, and portfolio management

Use area paths to group work items by product, feature, or business areas and to support teams responsible for work assigned to those areas. You can define a hierarchical set of area paths or a flat set. Typically, you define a hierarchical set of area paths to support a business hierarchy that wants to track progress of several teams.

Area paths and hierarchical grouping

The two main ways to group work items are by area path and by parenting them under a portfolio WIT as described early in this article. The two aren't mutually exclusive. Here are their differences:

- Area paths assigned to a team determine what work items appear in a team view: product backlog, portfolio backlog, delivery plans, or other portfolio planning tool
- Grouping work items under a parent feature or epic determine what rollup views are supported and how work appears in a portfolio planning tool

You can also assign tags to work items to group them for query and filter purposes. So when you structure your teams and projects, make sure you understand how you use these grouping tools to support your business needs. Your choices affect the use of portfolio planning tools.

Area path-dependent tools

To perform the following tasks, you must define area paths:

- Query and chart work items based on Area Path
- Assign work to more than one team
- Work with management and feature teams
- Filter a backlog, query, board, or plan using Area Paths

> **Tip**
>
> You can define your area path structure and assign area paths to teams. Or, you can add a team and create the area path with the team name at that time. If teams are fully independent, create a flat set of area paths. However, if you want to create a hierarchy of teams, then you'll want to create a tree-hierarchy of area paths. For more information, see **Configure a hierarchy of teams**.

To use the following tools, teams must subscribe to area paths:

- Boards> all tools
- Backlogs> all tools
- Sprints> all tools
- Delivery plans
- Rollup
- Analytics> all

Area paths and team assignments

Each project has a default team and default area path. In some cases, there's only one team to plan and track work. As organizations grow, however, you might add more teams to manage the backlog and sprints.

The following example shows area paths and their assignments to teams, which support portfolio management views for the Account Management and Service Delivery teams.

Areas	Teams
Fabrikam Fiber	
Account Management	Account Management
Customer Profile	Account Management, Customer Profile
Shopping Cart	Account Management, Shopping Cart
Service Delivery	Service Delivery
Email	Email, Service Delivery
Internet	Internet, Service Delivery
Service Status	Service Delivery, Service Status
TV	Service Delivery, TV
Voice	Service Delivery, Voice

For more information, see the following articles:

- Portfolio management
- About area paths
- About teams and Agile tools
- Agile culture.

Recommendations:

- Consider what upper management needs to know and how to best support them
- Consider how you want to use rollup both for a team and portfolio management
- Define epics and scenarios for large initiatives that take two or more sprints to complete
- Create hierarchical area paths to support sub categories of features and product areas
- Define requirements for work that can be accomplished in a single sprint and can be assigned to a single individual
- Define tasks to track more granular details or when you want to track time spent working

Tip

- You can only assign a work item to a single individual. When you define work items, consider how many work items you need and who to assign them to.
- Choose the `Node Name` field as a column option to view the leaf area node in a backlog list or board card. For more information, see **Customize cards**.
- Don't create parent-child links among work items of the same type, such as story-story, bug-bug, task-task.

Most Azure Boards tools support a filtered view of work items based on area path or iteration path. You can also apply more filters based on keyword, assignment, WIT, and more.

Bugs as requirements or tasks

Each team can choose how they want to manage bugs. Some teams like to track bugs along with requirements on the backlog. Other teams like to track bugs as tasks performed in support of a requirement. The bugs then appear on their Taskboard.

If you use the Scrum process, your default setup is to track bugs along with product backlog items (PBIs). If you work in a project based on the Agile or CMMI processes, bugs don't automatically appear on your backlog.

Determine with your team how you want to manage bugs. Then, change your team settings accordingly.

> **Tip**
>
> After you refresh a backlog or board and you don't see bugs where you expect them, review **How backlogs and boards display hierarchical (nested) items**. Only leaf nodes of nested items appear on sprint Taskboards.

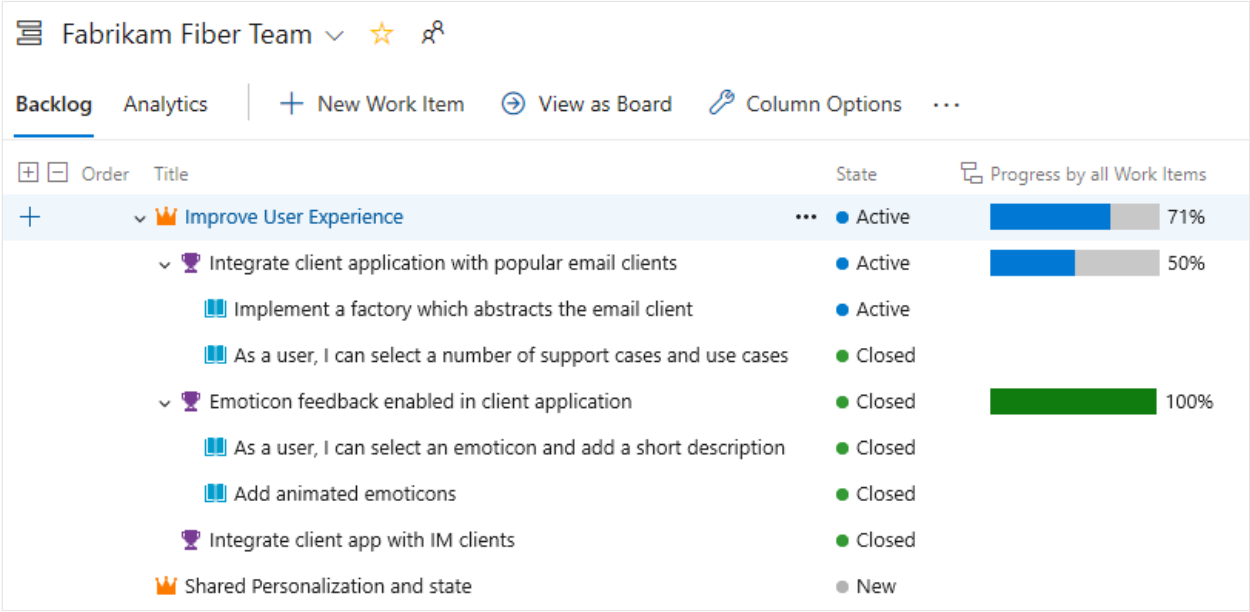
System WITs on a backlog

To track issues or impediments along with your requirements or in a portfolio backlog, add them to your custom Inherited process. For more information, see Customize your backlogs or boards (Inheritance process).

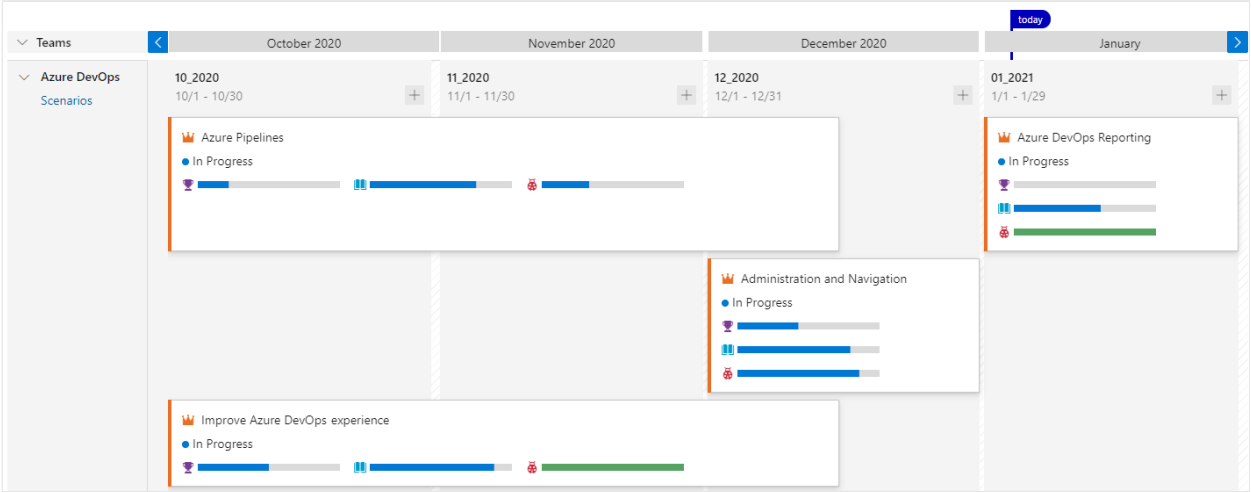
Rollup, hierarchy, and portfolio management

Rollup columns allow you to view progress bars or totals of numeric fields or descendant items within a hierarchy. Descendant items correspond to all child items within the hierarchy. You can add one or more rollup columns to a product or portfolio backlog.

Here we show **Progress by all Work Items**, which displays progress bars for ascendant work items based on the percentage of closed descendant items.



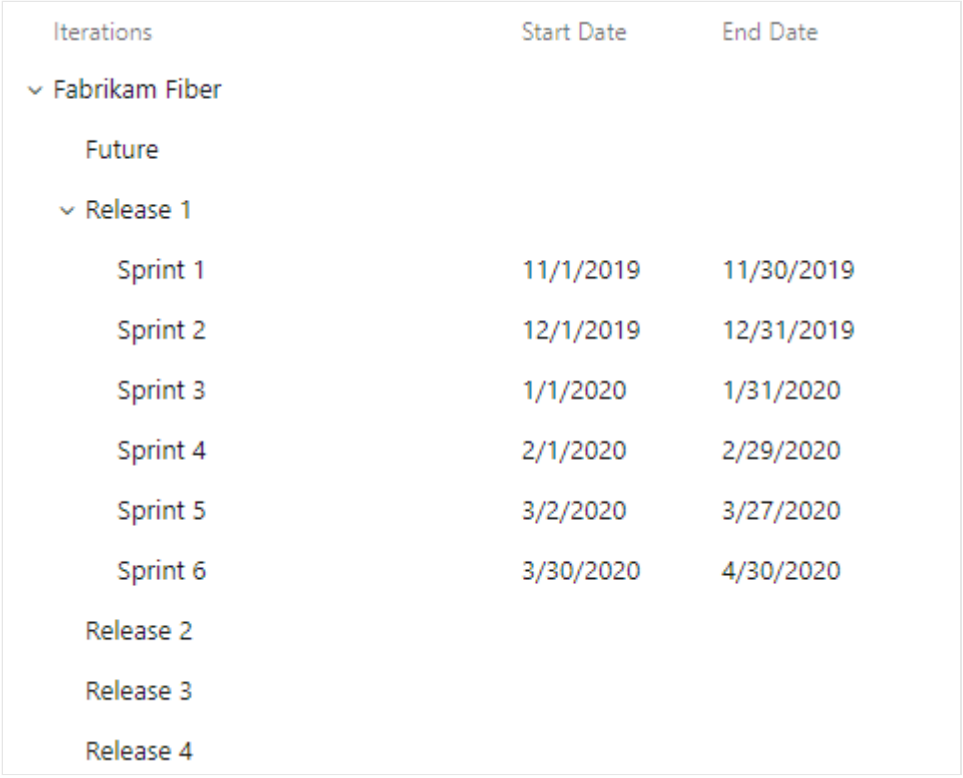
Delivery Plans supports rollup views of epics, features, and other custom portfolio backlogs.



Iteration paths sprints releases & versioning

Iteration paths support Scrum and Scrumban processes where work is assigned to a set time period. Iteration paths allow you to group work into sprints, milestones, or other event-specific or time-related period. Each iteration or sprint corresponds to a regular time interval referred to as a sprint cadence. Typical sprint cadences are two weeks, three weeks, or a month long. For more information, see About area and iteration paths.

Iteration paths can be a flat list, or grouped under release milestones as shown in the following image.



Iterations	Start Date	End Date
Fabrikam Fiber		
Future		
Release 1		
Sprint 1	11/1/2019	11/30/2019
Sprint 2	12/1/2019	12/31/2019
Sprint 3	1/1/2020	1/31/2020
Sprint 4	2/1/2020	2/29/2020
Sprint 5	3/2/2020	3/27/2020
Sprint 6	3/30/2020	4/30/2020
Release 2		
Release 3		
Release 4		

While Iteration Paths don't affect Kanban board tools, you can use Iteration Paths as a filter on boards. For more information, see Filter your Kanban board.

Define iteration paths and assign them to teams when you want to use the following tools:

- Assign work items to sprints using the Planning pane
- Query and chart work items based on Iteration Path
- Forecast your product backlog
- Sprints> all tools
- Delivery Plans, a calendar view of team deliverables
- Velocity chart and Sprint burndown chart

> **Tip**
>
> If a team hasn't subscribed or selected an iteration path, that iteration path won't appear in a team view or tool.

Time tracking

Most organizations following Scrum processes use time estimates for Sprint capacity planning. Azure Boards tools fully support tracking time for this purpose. The main field

used is the task `Remaining Work` field, which typically zeros out at the end of the sprint.

However, some organizations require time tracking to support other purposes, such as for billing or maintaining time allocation records. Time values for estimated work and completed work are of interest. The Agile and CMMI processes provide these fields—Original Estimate, Completed Work, Remaining Work—for use in tracking time. You can use them for that purpose. However, Azure Boards provides limited native support for time tracking. Instead, consider using a Marketplace extension to support your other time tracking requirements.

> **Note**
>
> The `Original Estimate`, `Completed Work`, `Remaining Work` fields were designed to support integration with Microsoft Project. Integration support with Microsoft Project is deprecated for Azure DevOps Server 2019 and later versions, including the cloud service.

Process changes that affect all teams

Any change made to a process in a project impacts all teams in that project. Many changes don't cause much disruption to the teams they support, but the following changes do.

Custom fields

When you add custom fields to a WIT, it doesn't directly affect any specific tool. Instead, these fields become visible within the corresponding work items. For instance, if you introduce a custom numeric field, you can utilize it for rollup calculations on backlogs. Also, you can use this custom field with the following reporting tools. So, while the effect isn't tool-specific, it does enhance your ability to tailor work items to your project's needs.

- In-context Velocity report and dashboard widget
- In-context Sprint Burndown report and dashboard widget
- Dashboard Burndown and Burnup widget

> **Note**

All default and custom fields are shared across all projects in a collection or organization. There is a limit of 1024 fields that you can define for a process.

Custom WITs

The following table shows the effects when you add a custom WIT to a specific category.

Task

Requirement

Epic or feature

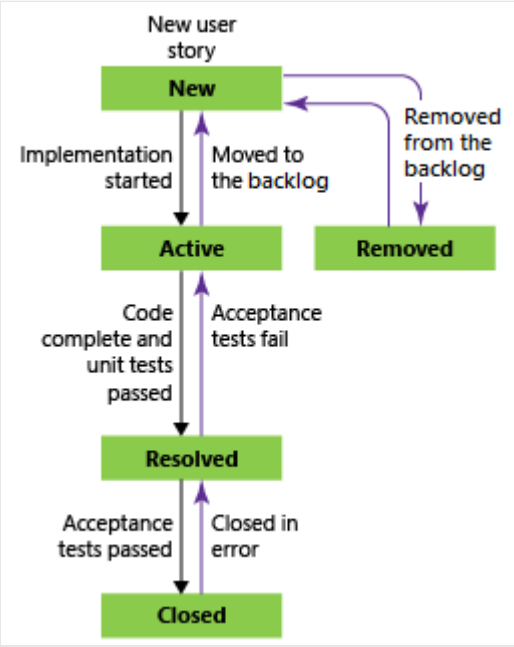
- Child work items of the new WIT appear on the product backlog
- Work items based on the new WIT appear on the sprint backlogs and Taskboards
- Work items based on the new WIT appear on the product backlog and Kanban board
- Each team must configure the Kanban board to support the new WIT
- Work items based on the new WIT appear on the corresponding portfolio backlogs and Kanban boards
- Each team must configure the Kanban boards to support the new WIT
- The new WITs might not appear on one or more of the portfolio planning tools

Custom workflow

Each process supports a default workflow. This workflow defines the default columns that appear on the Kanban boards and sprint Taskboards.

Agile process

Workflow states: User Story, Agile process



Sometimes, teams want to track the status of their work that goes beyond these default states. Support is provided in one of the following ways:

- Add custom workflow states to the WIT: This option impacts all teams and requires that they update their Kanban board configuration.
- Add columns to a Kanban board: This option only impacts the team that adds the columns.

Both workflow states and Kanban columns appear in the Cumulative Flow diagram for a team. Individuals can choose which columns appear in the chart. For more information, see Cumulative flow diagram.

Who can make changes?

Since process-level, project-level, and team-level settings can have a wide effect, changes are restricted to users with the following required permissions.

Process-level changes

To create, edit, or manage Inherited processes and apply them to projects, you must be a member of the **Project Collection Administrators** group. Or, you must have the corresponding permissions **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**. See Set permissions and access for work tracking, Customize an inherited process.

For more information, see the following articles:

- About the inheritance process
- Customize a project
- Create and manage a process

Project-level changes

To add Area Paths or Iteration Paths, you must be a member of the **Project Administrators** group.

Or, to add, edit, and manage Area Paths or Iteration Paths under a specific node, you must have one or more of the following permissions set to **Allow**:

- **Create child nodes**
- **Delete this node**
- **Edit this node**
- **View permissions in this node**

For more information, see the following articles:

- Define area paths & assign to a team
- Define iteration paths (sprints) & assign team iterations

Team-level changes

To configure team tools, you must be a team administrator or a member of the **Project Administrators** group.

Team administrators do the following operations:

- Add team members
- Subscribe to area and iteration paths
- Configure backlogs and other common team settings
- Configure Kanban boards
- Manage team notifications

For more information on configuring backlogs and boards, see Manage and configure team tools.

Next steps

Get started as an administrator

Related articles

- Configuration and Customization FAQs
- Set up your Backlogs and Boards
- Inherited process model
- Manage and configure team tools

Feedback

Was this page helpful? Yes No

Provide product feedback

Connect Azure Boards to GitHub (cloud)

Article • 08/22/2024

Azure DevOps Services

Use GitHub.com repositories for your software development and your Azure Boards project to plan and track your work. Connect your project and repo so your GitHub commits and pull requests get linked to your work items in Azure Boards.

> **Note**
>
> Azure Boards and Azure DevOps Services support integration with GitHub.com and GitHub Enterprise Server repositories. If you want to connect from an on-premises Azure DevOps Server, see Connect Azure DevOps Server to GitHub Enterprise Server.

Prerequisites

- **Project requirements:** Have an Azure DevOps project. If you don't have a project, create one.
- **Permissions:**
  - Be a member of the **Project Administrators** group. If you created the project, you have permissions.
  - Be an administrator or owner of the GitHub repository to connect to. You can connect to multiple GitHub repositories as long as you're an administrator for those repositories.

Authentication options

The following authentication options are supported based on the GitHub platform you want to connect to.

GitHub.com

GitHub Enterprise Server

- GitHub.com user account (Recommended)
- Personal access token (PAT)
- OAuth (preferred, registration required)

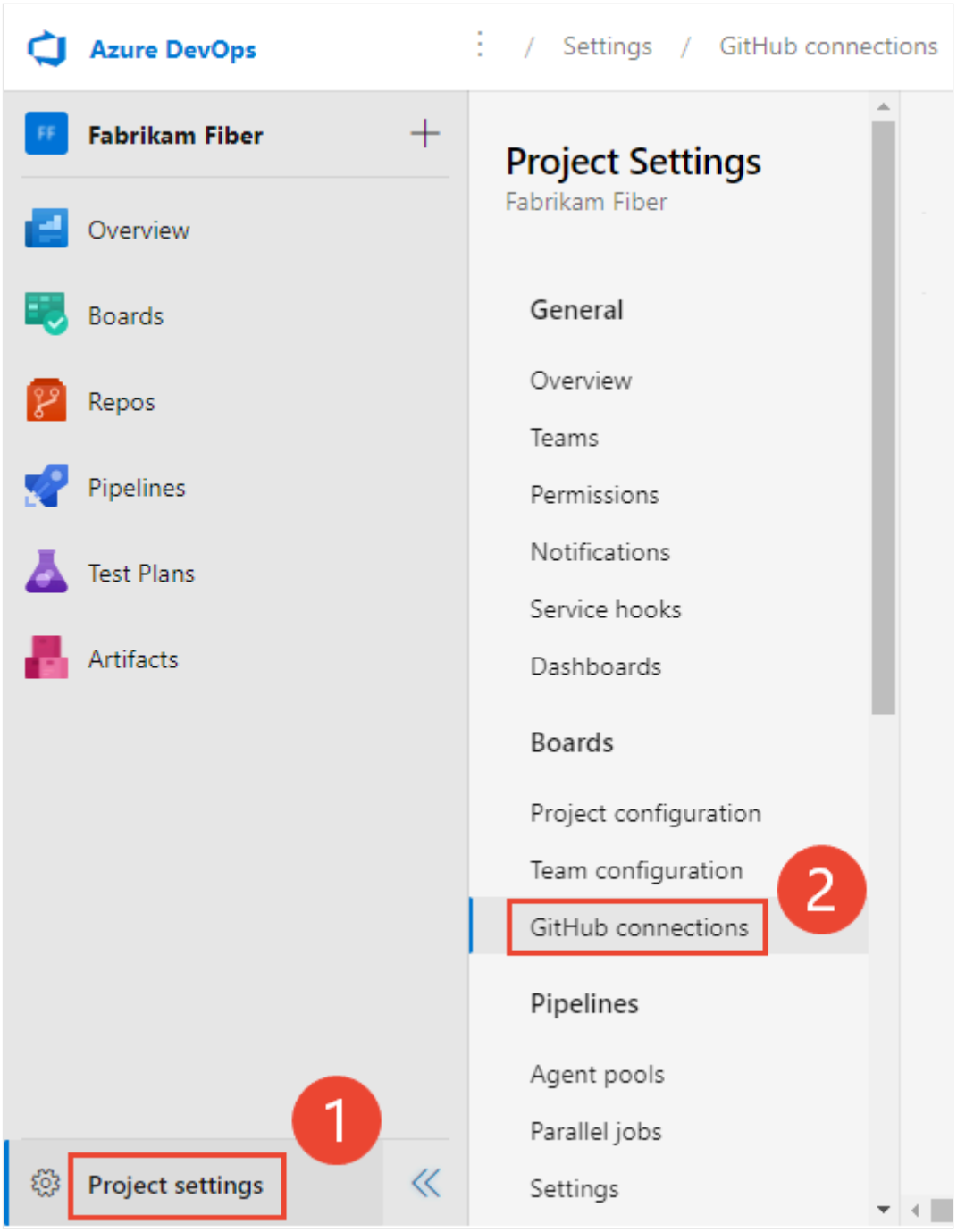
- PAT
- Username plus password

> **Note**
>
> If you choose to connect GitHub with a PAT, configure single sign-on (SSO) for the PAT on your GitHub account. This configuration is necessary to retrieve a list of repositories from an organization with Security Assertion Markup Language (SAML) SSO authentication enabled.

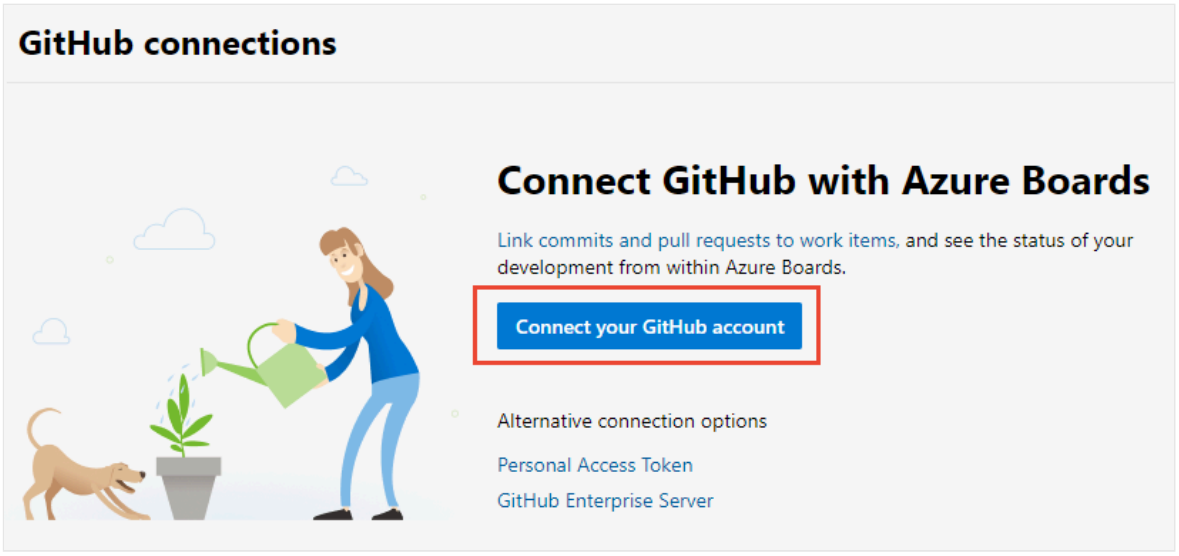
Connect Azure Boards to a GitHub repo.

1. Sign in to your project (`https://dev.azure.com/{Your_Organization/Your_Project}`).

2. Select **Project settings** > **GitHub connections**.



3. If this is your first time connecting from the project, select **Connect your GitHub account** to use your GitHub account credentials.



Otherwise, choose + **New connection**, and select your authentication method from the **New Connection** dialog.

When you connect with your GitHub account, use your GitHub credentials to authenticate. To use a PAT, see Add a GitHub connection using PAT. For connecting to a GitHub Enterprise Server, see Register Azure DevOps in GitHub as an OAuth App.

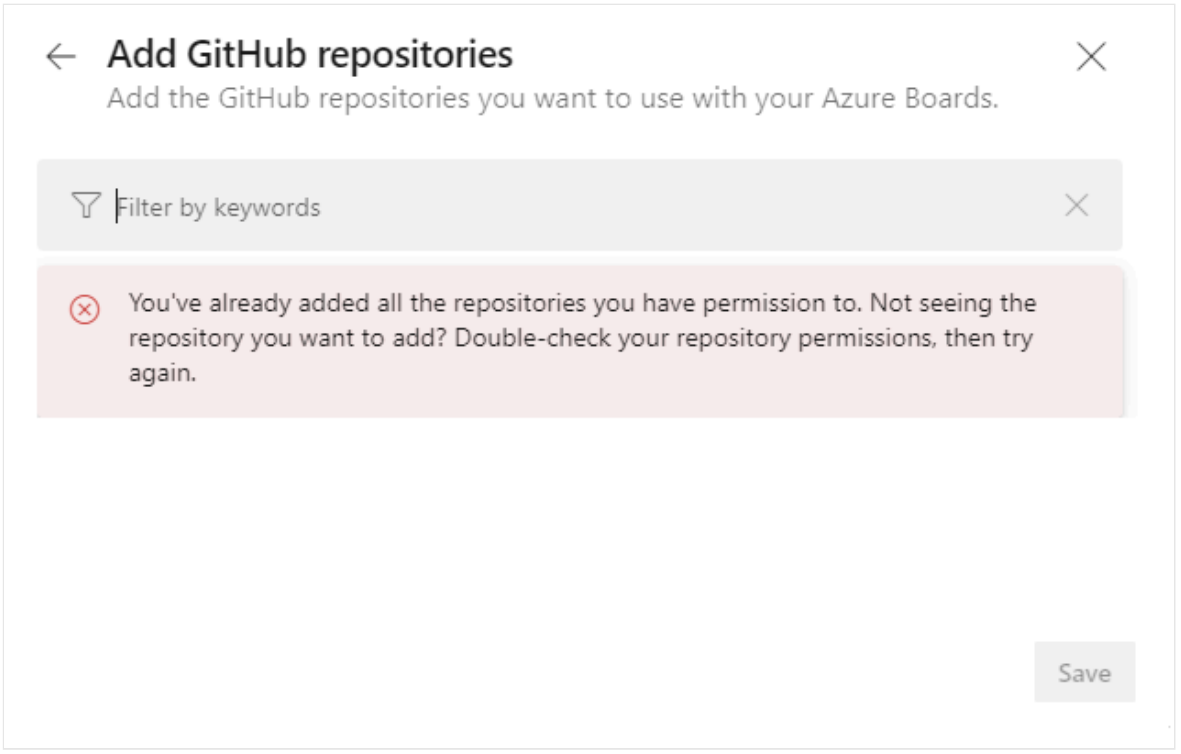
Add a GitHub connection with GitHub credentials

You can connect up to 500 GitHub repositories to an Azure Boards project.

1. If this is your first time connecting to GitHub from Azure Boards, sign in using your GitHub credentials. Choose an account for which you're a repository administrator.

2. Select the GitHub account or organization you want to connect. Only organizations that you own or are an administrator for are listed.

   If all repositories for an organization are already connected to Azure Boards, you will see the following message:

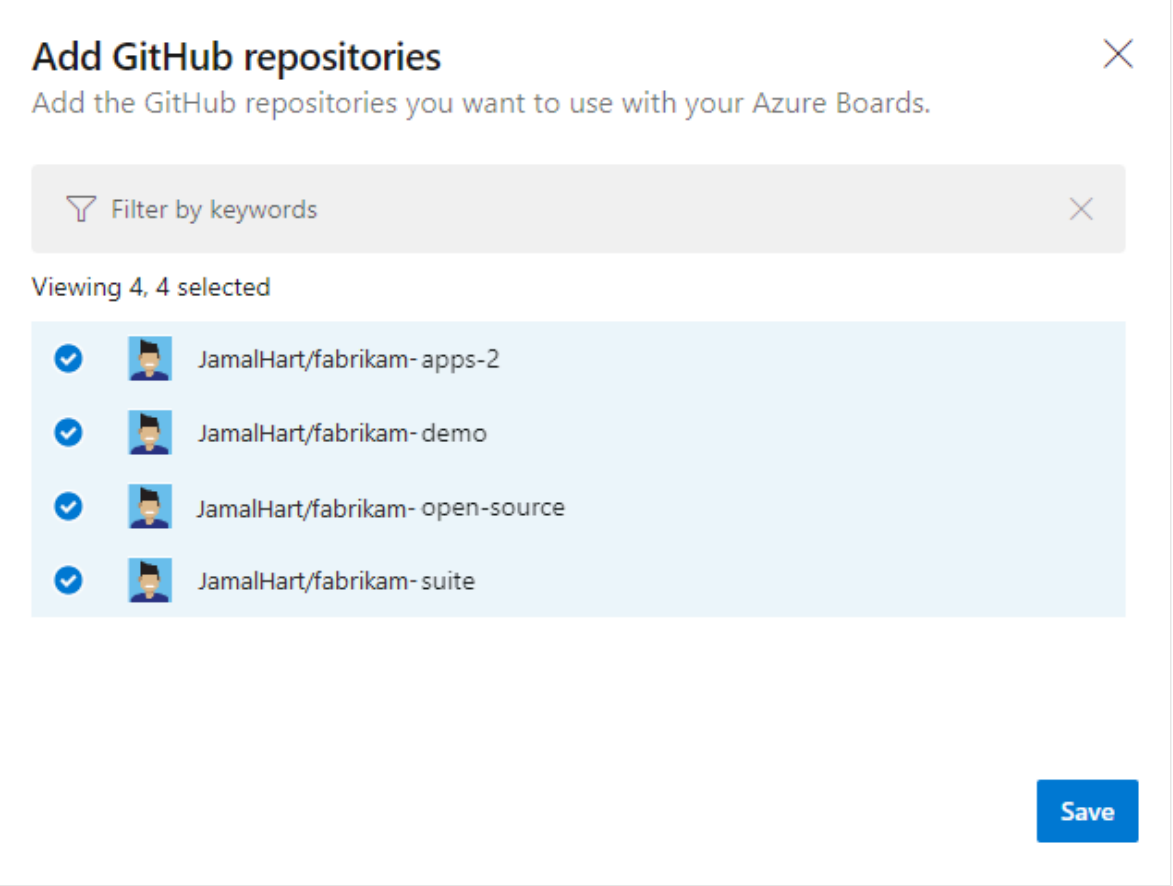


3. Enter your GitHub credentials. If you have two-factor authentication enabled, enter the authentication code sent by GitHub and select **Verify**. If not, the system will automatically recognize your GitHub organization as your GitHub account is associated with your Azure DevOps Services account.

Add GitHub repositories

Once authenticated, you can select the repositories you want to connect.

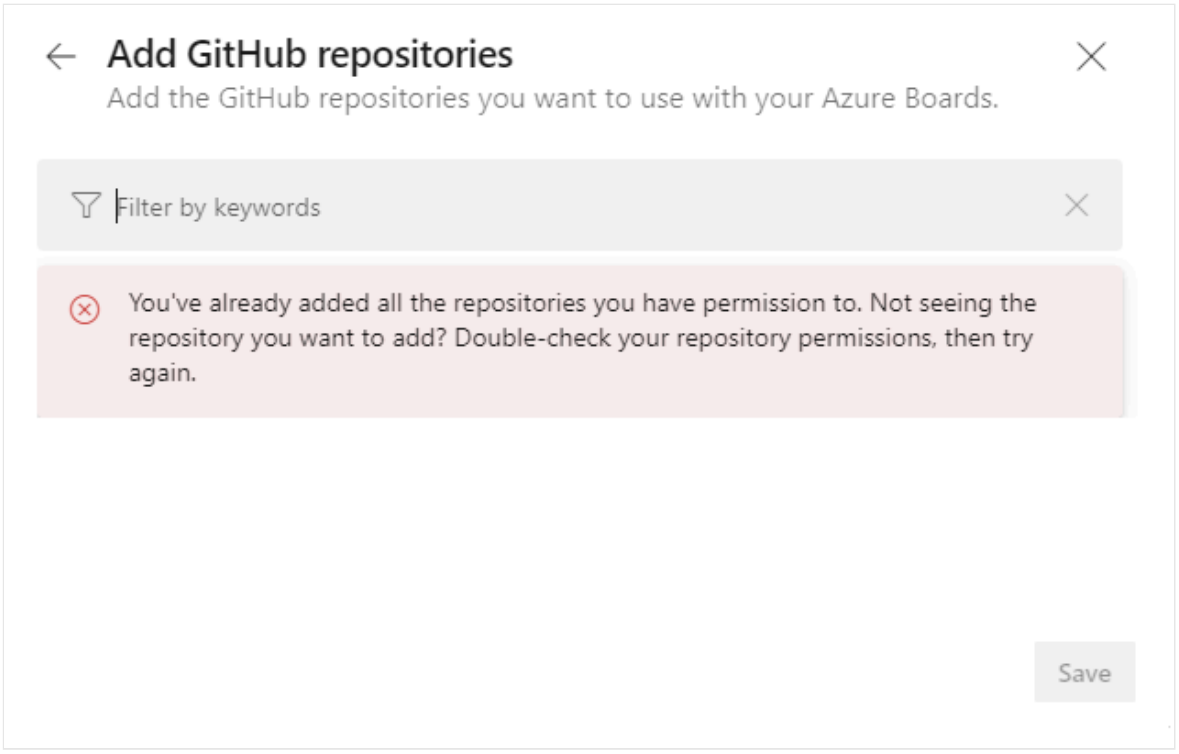
1. The **Add GitHub Repositories** dialog automatically displays and selects all GitHub.com repositories for which you're an administrator in the selected organization. Deselect any repositories that you don't want to include in the integration.



Tip

To avoid unexpected **AB#** mention linking, we recommend connecting a GitHub repo to projects within a single Azure DevOps organization. Connecting the same GitHub repo to projects in multiple Azure DevOps organizations can cause issues. For more information, see Troubleshoot GitHub & Azure Boards integration.

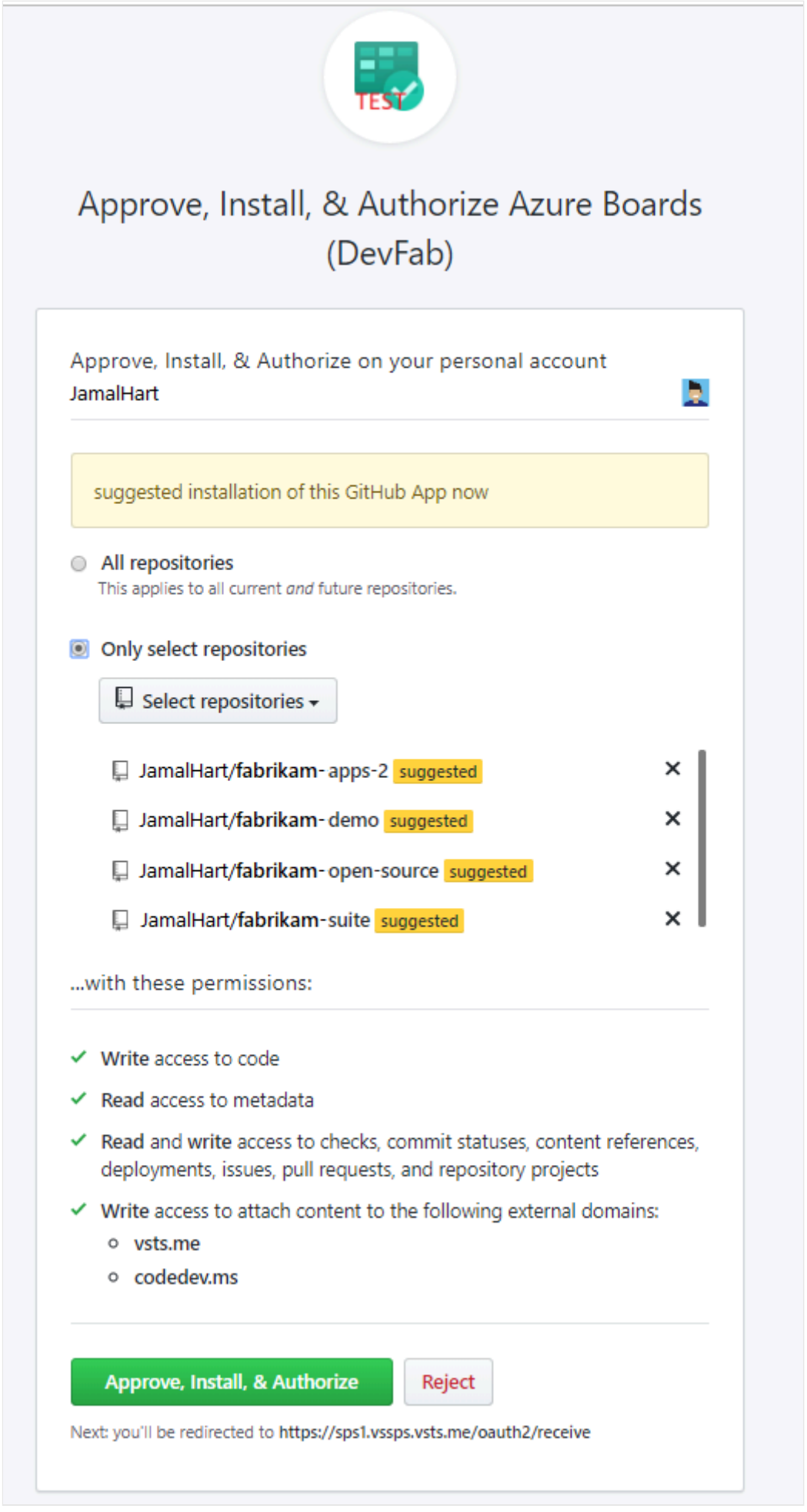
If all repositories are already connected to the current or another organization, the following message displays:



2. After making your selections, select **Save**.

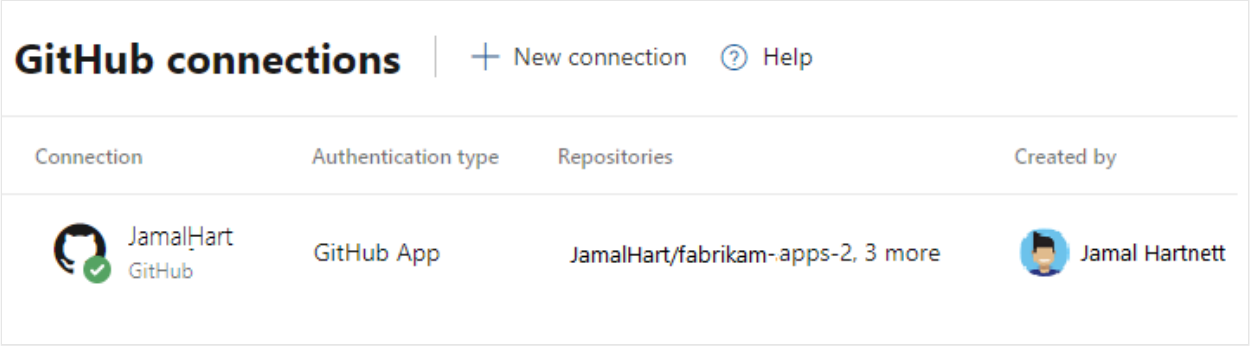
Confirm the connection

1. Review the GitHub page that displays and then choose **Approve, Install, & Authorize**.



2. Provide your GitHub password to confirm.

3. When you're done, you should see the new connection with the selected repositories listed.



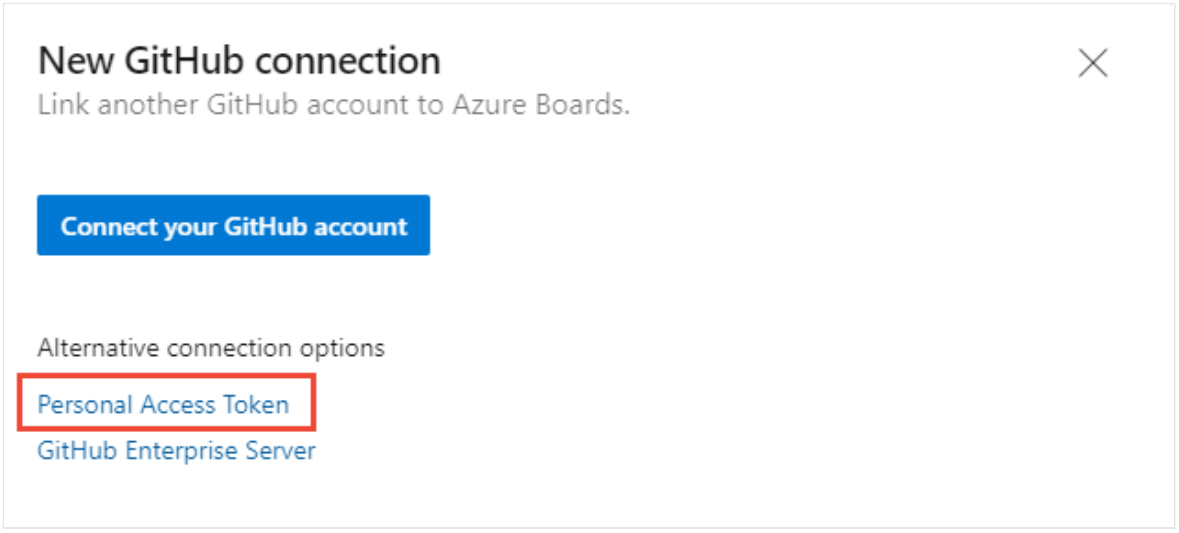
To change the configuration or manage the Azure Boards app for GitHub, see Change repository access to Azure Boards.

Add a GitHub connection using PAT

We recommend using your GitHub account credentials to connect to your GitHub repository. If you need to use a PAT, do the following steps.

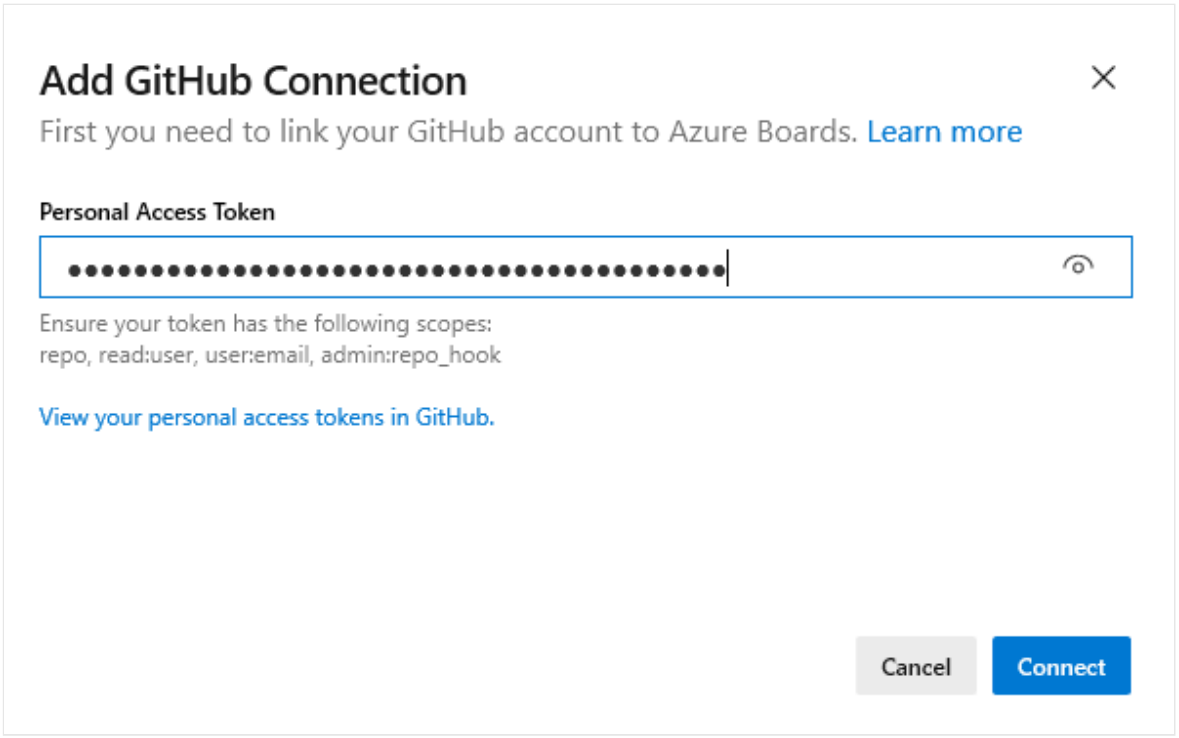
> **Tip**
>
> When you create your GitHub PAT, make sure that you include these scopes: `repo, read:user, user:email, admin:repo_hook`.

1. Choose **Personal Access Token**.



   To create a GitHub PAT, go to GitHub Developer Settings > Personal access tokens.

2. Enter the PAT and select **Connect**.



3. Select the repositories you want to connect to the project by following the procedures outlined in Choose the repositories earlier in this article.

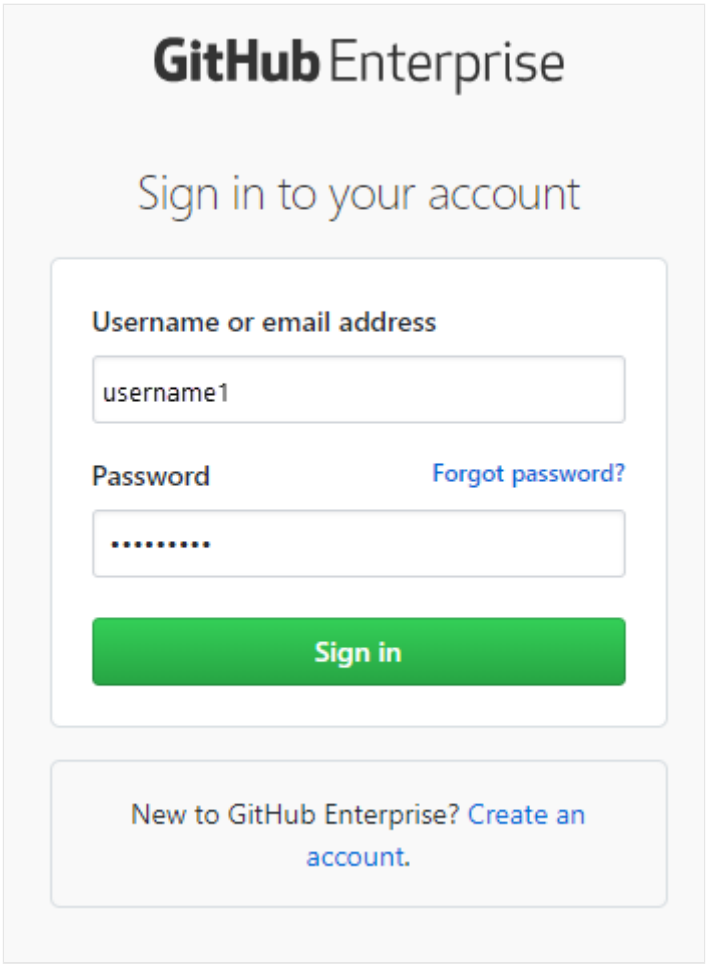
4. If it's the first time connecting to a GitHub account or organization from Azure Boards, install the Azure Boards app for GitHub. Confirm the connection earlier in this article.

Register Azure DevOps in GitHub as an OAuth App

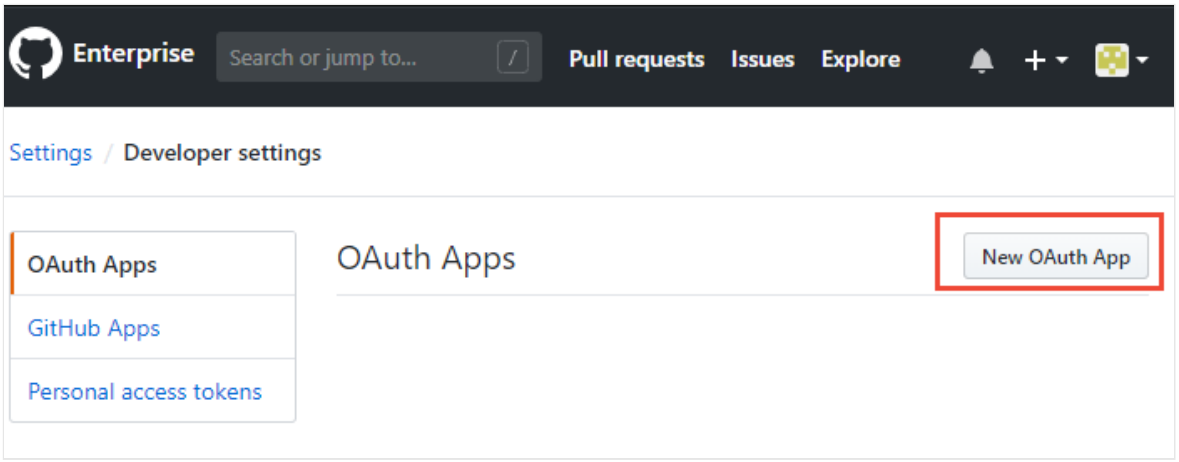
To use OAuth to connect Azure DevOps with your GitHub Enterprise Server, first register the application as an OAuth App. For more information, see Create an OAuth App.

Register Azure DevOps Services

1. Sign into the web portal for your GitHub Enterprise server.



2. Open **Settings** > **Developer settings** > **Oauth Apps** > **New OAuth App**.



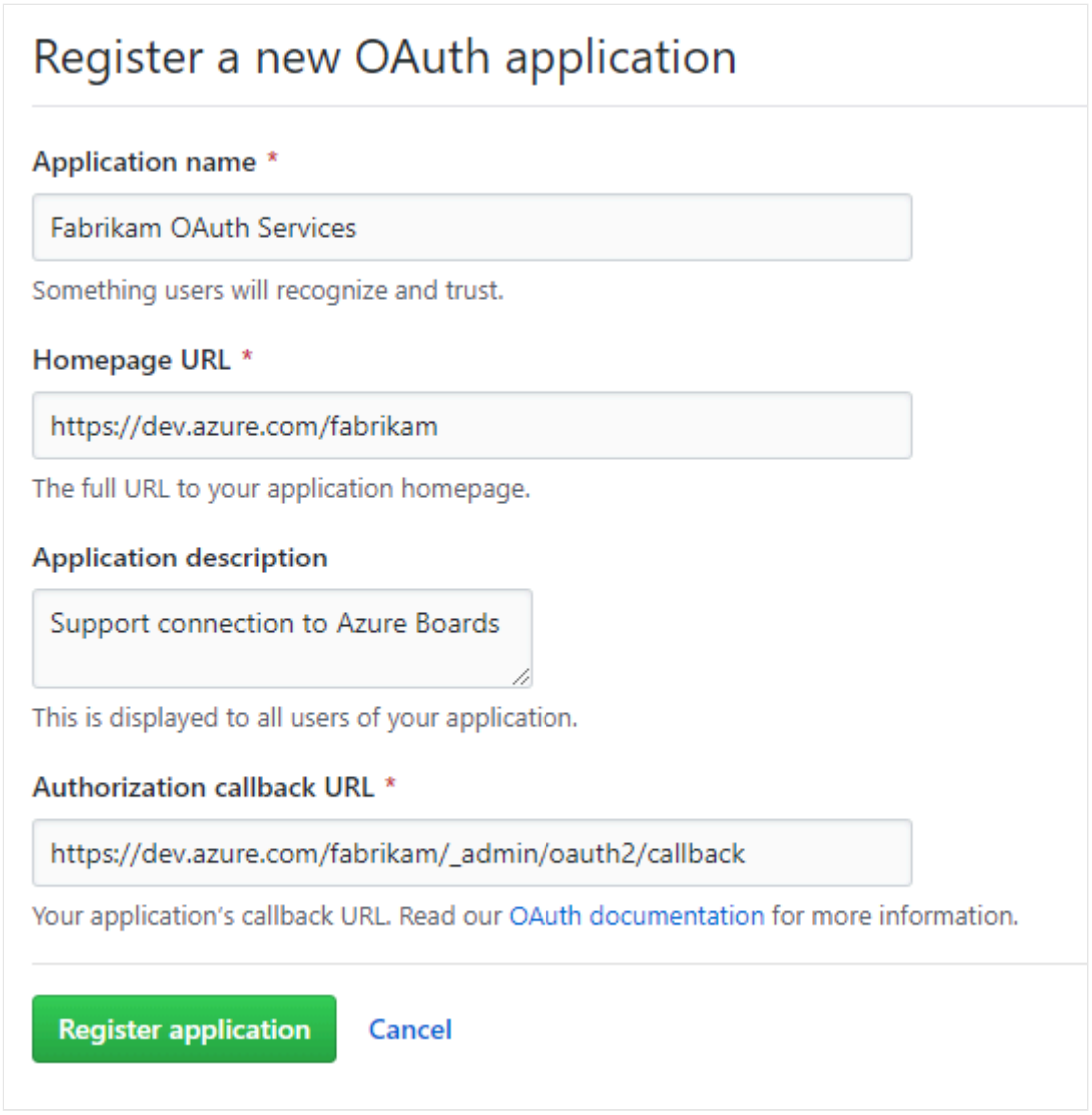
3. Enter registration information.

   For the **Homepage URL**, specify the **Organization URL** of your organization. For the **Authorization callback URL**, use the following pattern to construct the URL.

   ```
   {Azure DevOps Services Organization URL}/_admin/oauth2/callback
   ```

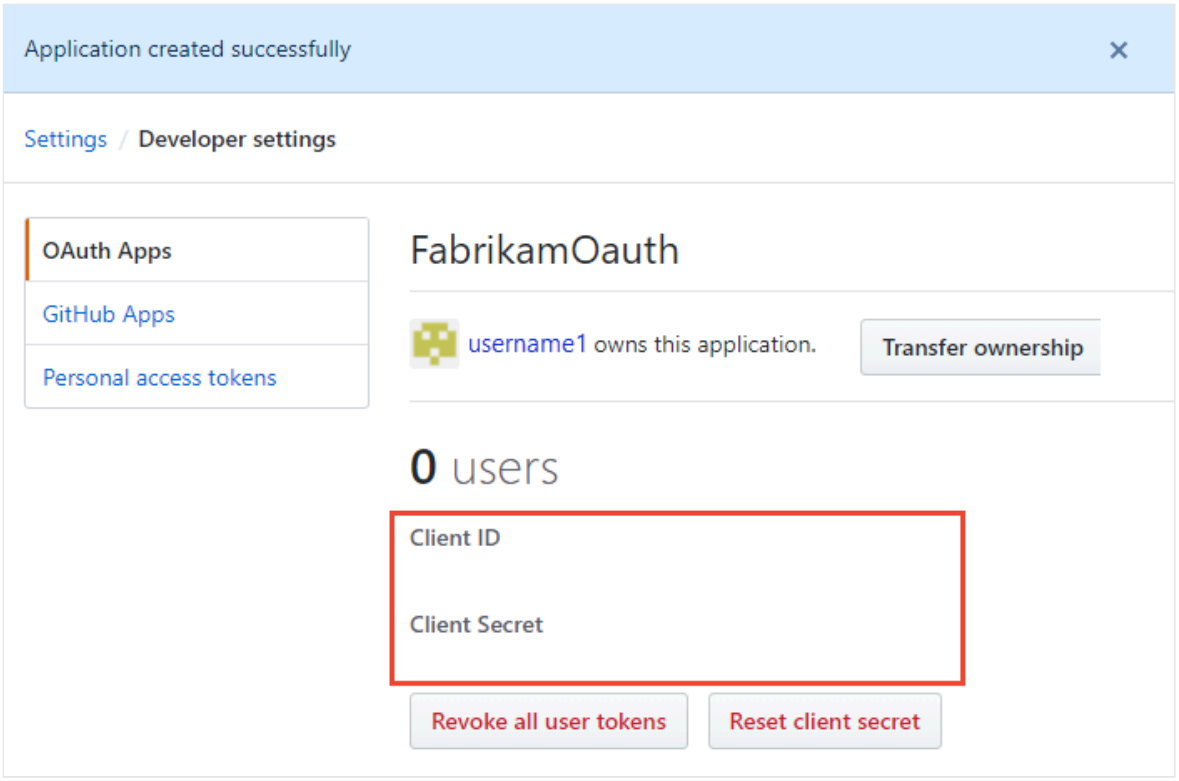
   For example:

   ```
   https://dev.azure.com/fabrikam/_admin/oauth2/callback
   ```



4. Select **Register application**.

5. The **Client ID** and **Client Secret** for your registered OAuth application appear.

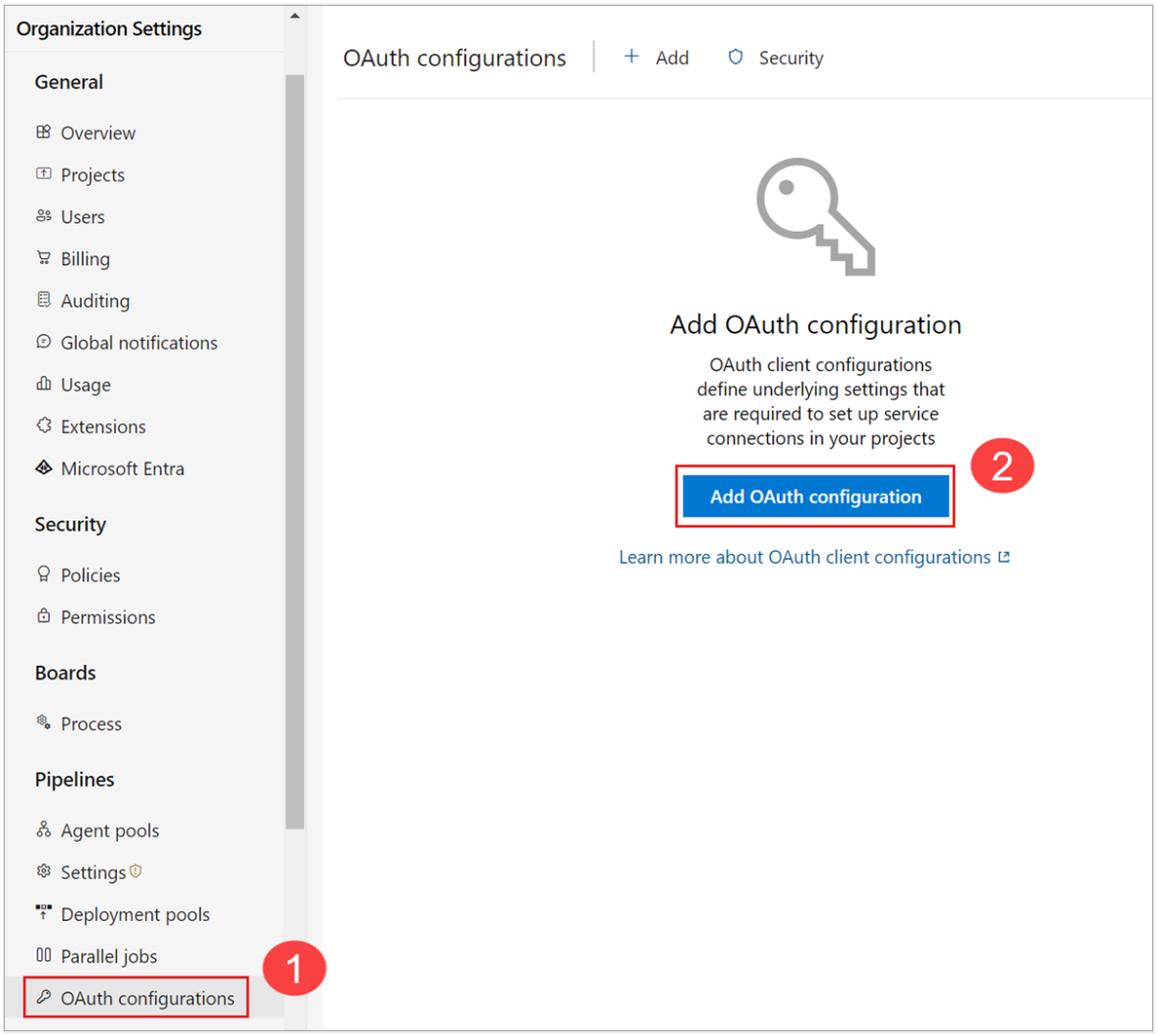


Register your OAuth configuration in Azure DevOps Services

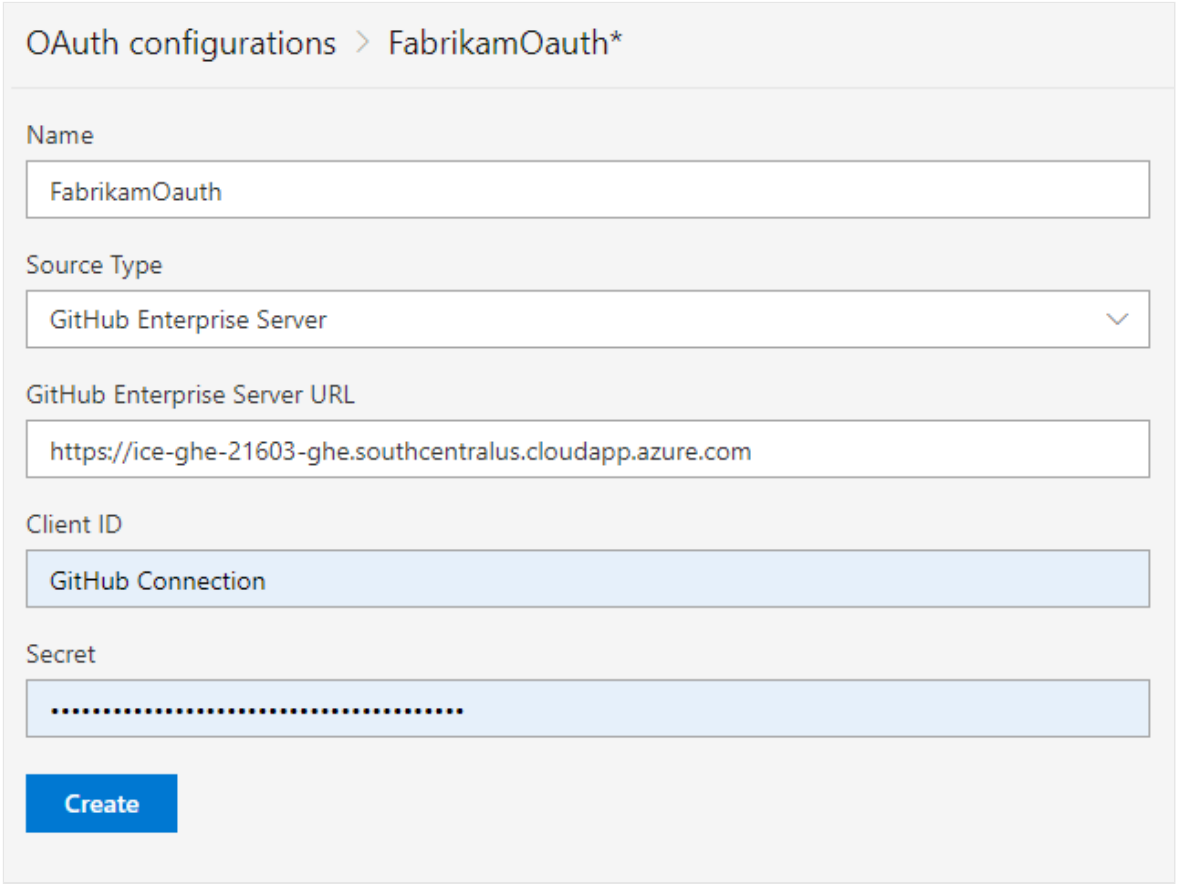
1. Sign into the web portal for Azure DevOps Services.

2. Add the GitHub Enterprise Oauth configuration to your organization.

3. In **Organization settings**, select **Oauth configurations** > **Add Oauth configuration**.



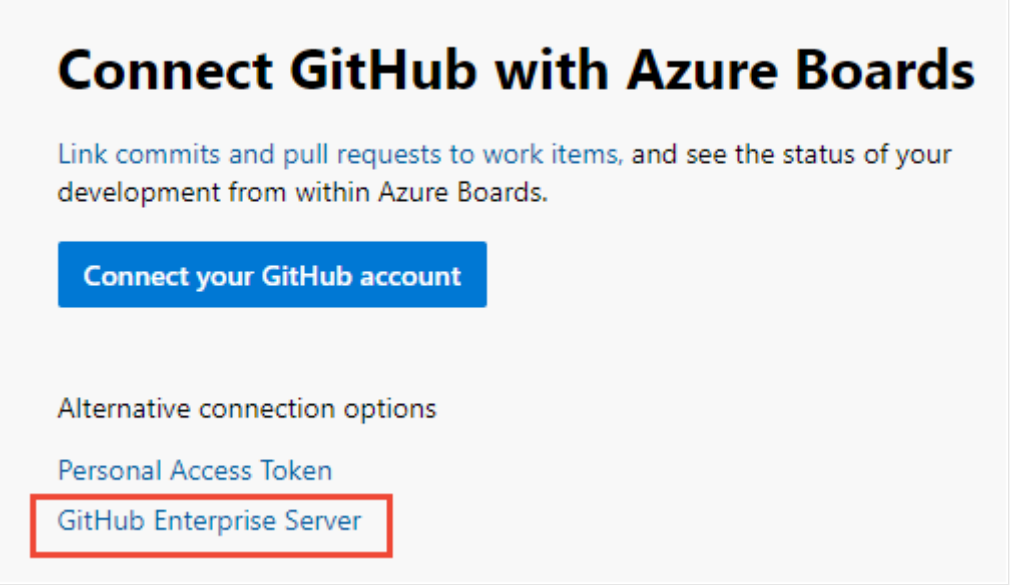
4. Enter your information, and then select **Create**.



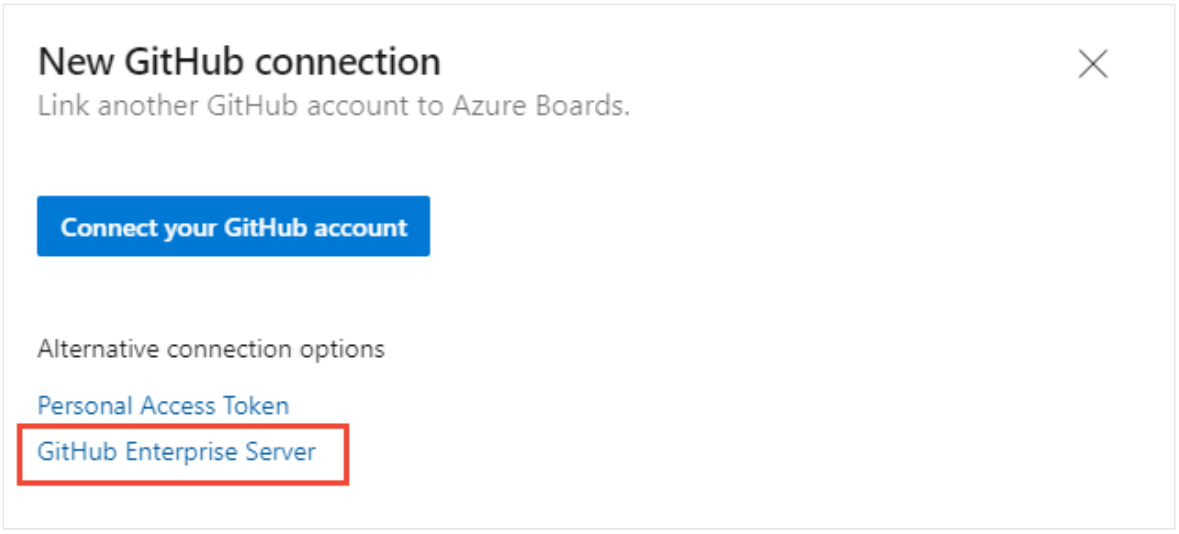
Connect Azure DevOps Services to GitHub Enterprise Server

> **Important**
>
> To connect Azure DevOps Services to your GitHub Enterprise Server, ensure your GitHub Enterprise Server is accessible from the Internet. Verify that Azure DNS can resolve your GitHub Enterprise Server name and that your firewall allows access from Azure Data Center IP addresses. To determine the IP address range, see Microsoft Azure Data Center IP Ranges. A common error message for connectivity issues is:
>
> *The remote name could not be resolved: 'github-enterprise-server.contoso.com'*
>
> If you encounter this error, check your server's accessibility. For more information, see Azure DNS FAQ.

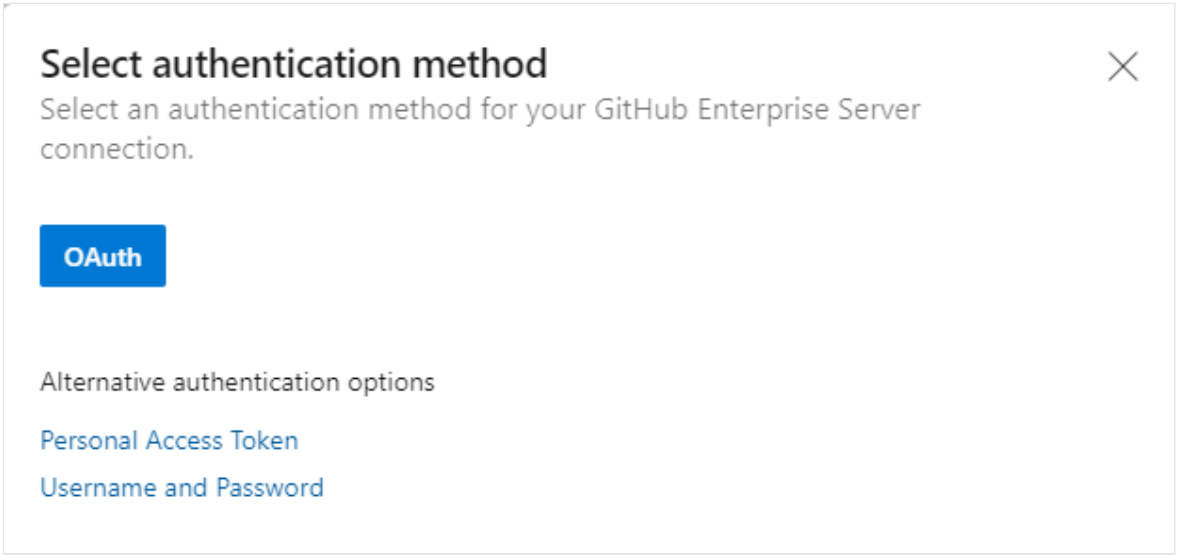
1. Select **Project settings** > **GitHub connections** > **GitHub Enterprise Server** for a first-time connection.



   Or, from the **New GitHub connection** dialog, select **GitHub Enterprise Server**.

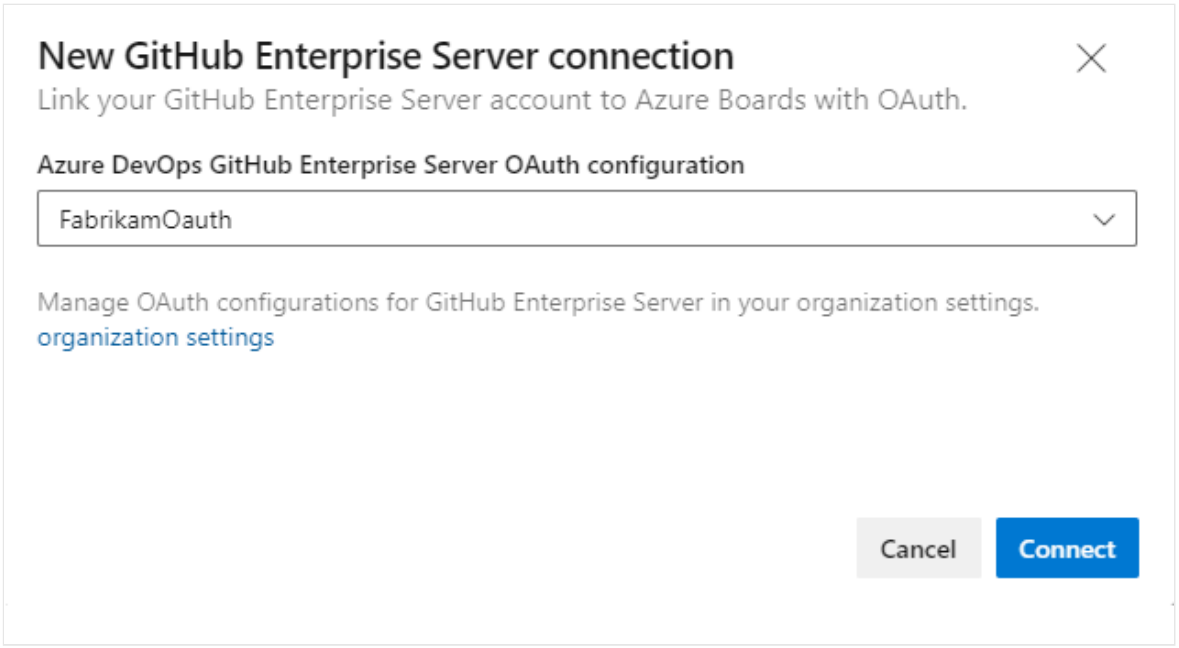


2. Select the authentication method.



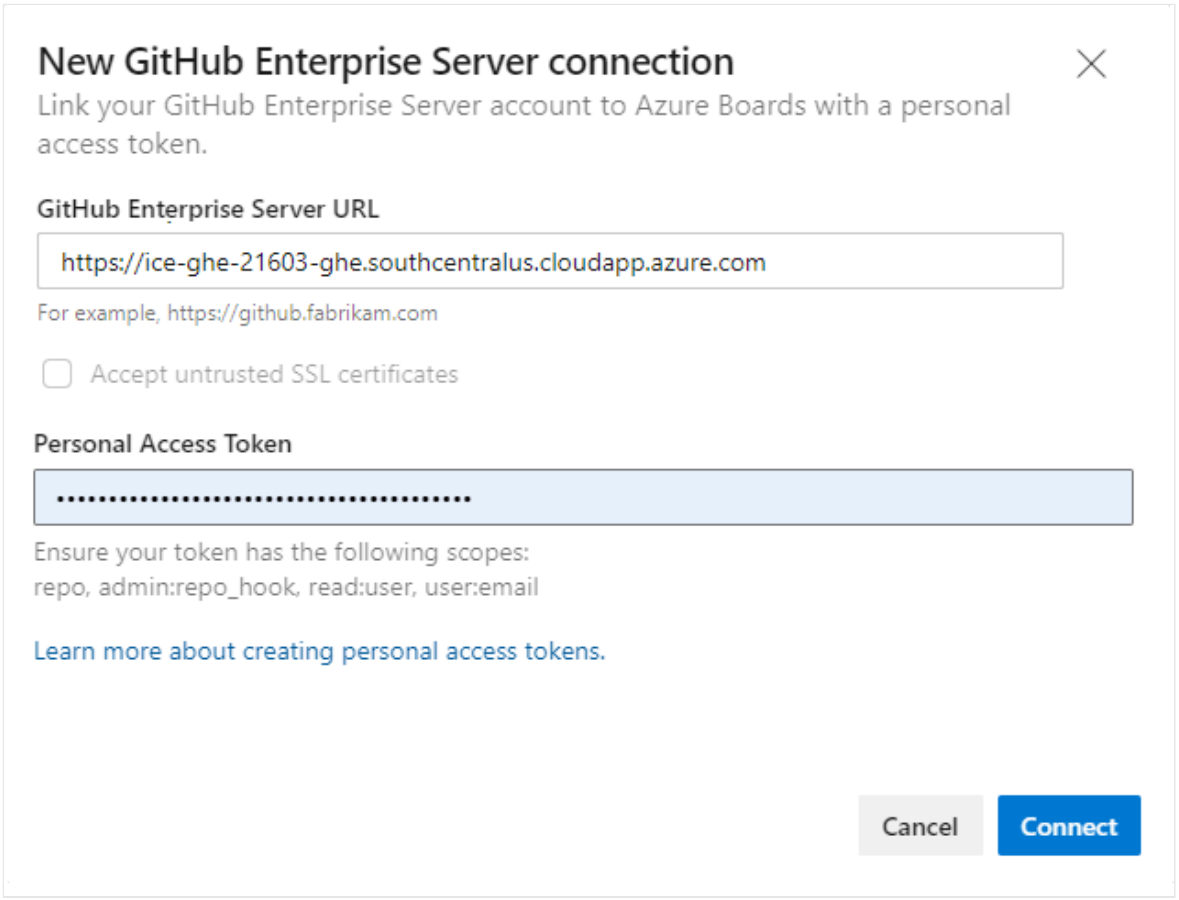
Connect with OAuth

Choose the configuration that you set up in Step 4 of Register your OAuth configuration in Azure DevOps Services, and then choose **Connect**.



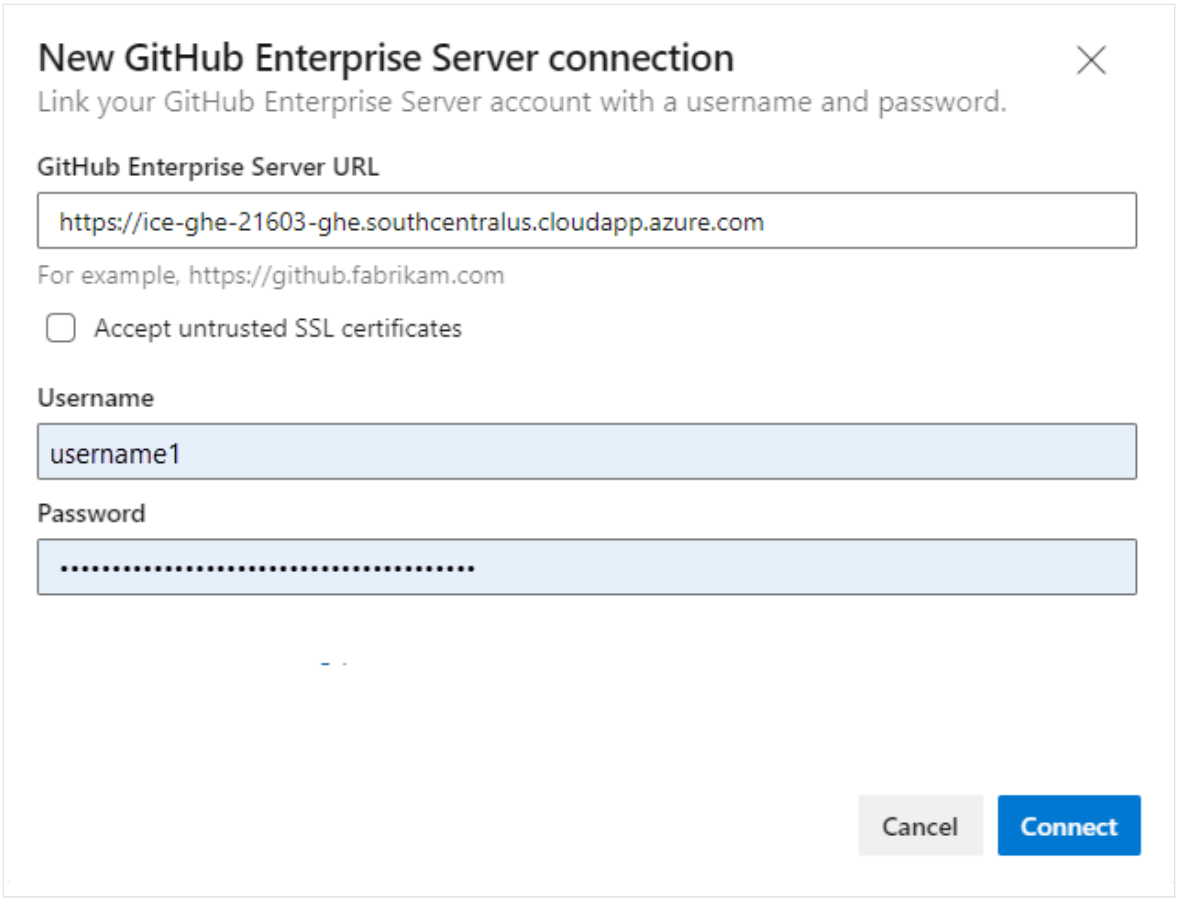
Connect with a Personal Access Token

Enter the URL for your GitHub Enterprise server and the Personal access token credentials recognized by that server. And then choose **Connect**.

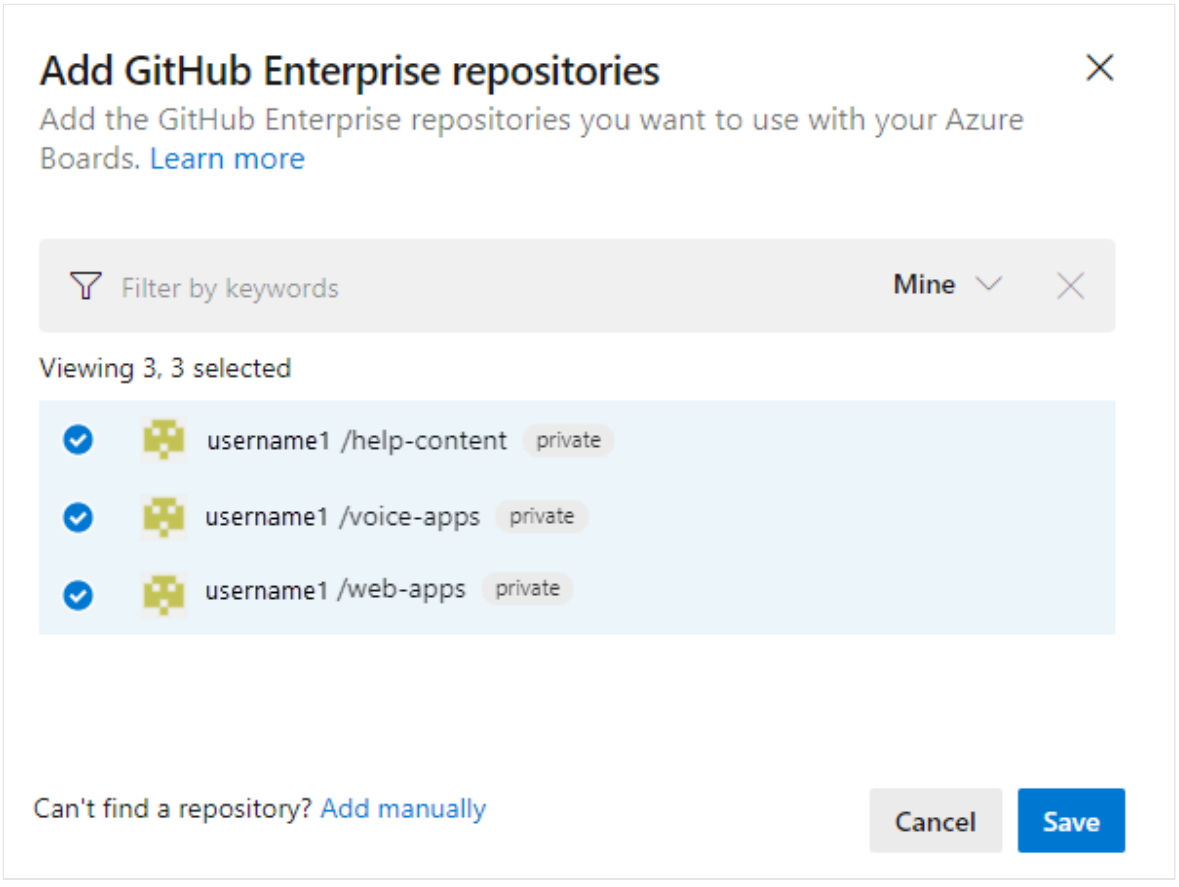


Connect with a Username and Password

Enter the URL for your GitHub Enterprise server and the administrator account credentials recognized by that server, and then select **Connect**.



3. The dialog lists all repositories for which you have GitHub administration rights. You can toggle between **Mine** and **All** to determine if others appear, and then check the ones that you want to add. Select **Save** when you're done.



> **Tip**
>
> You can only make a connection to repositories defined under one GitHub organization. To connect a project to other repositories defined in another GitHub organization, you must add another connection.

4. If it's your first time connecting to a GitHub account or organization from Azure Boards, you also install the Azure Boards app for GitHub. Confirm the connection earlier in this article.

Resolve connection issues

The Azure Boards-GitHub integration uses various authentication protocols to maintain the connection. Changes to a user's permission scope or authentication credentials can revoke the GitHub repositories connected to Azure Boards.

For an overview of the integration supported by the Azure Boards app for GitHub, see Azure Boards-GitHub integration.

Supported authentication options

The following supported authentication options depend on the GitHub platform you are connecting to:

Platform

GitHub.com

GitHub Enterprise Server

Azure DevOps Services

- GitHub.com user account
- Personal access token (PAT)
- OAuth
- PAT
- Username plus password

Azure DevOps Server 2020

Not applicable

- PAT

- Username plus password

Azure DevOps Server 2019

Not applicable

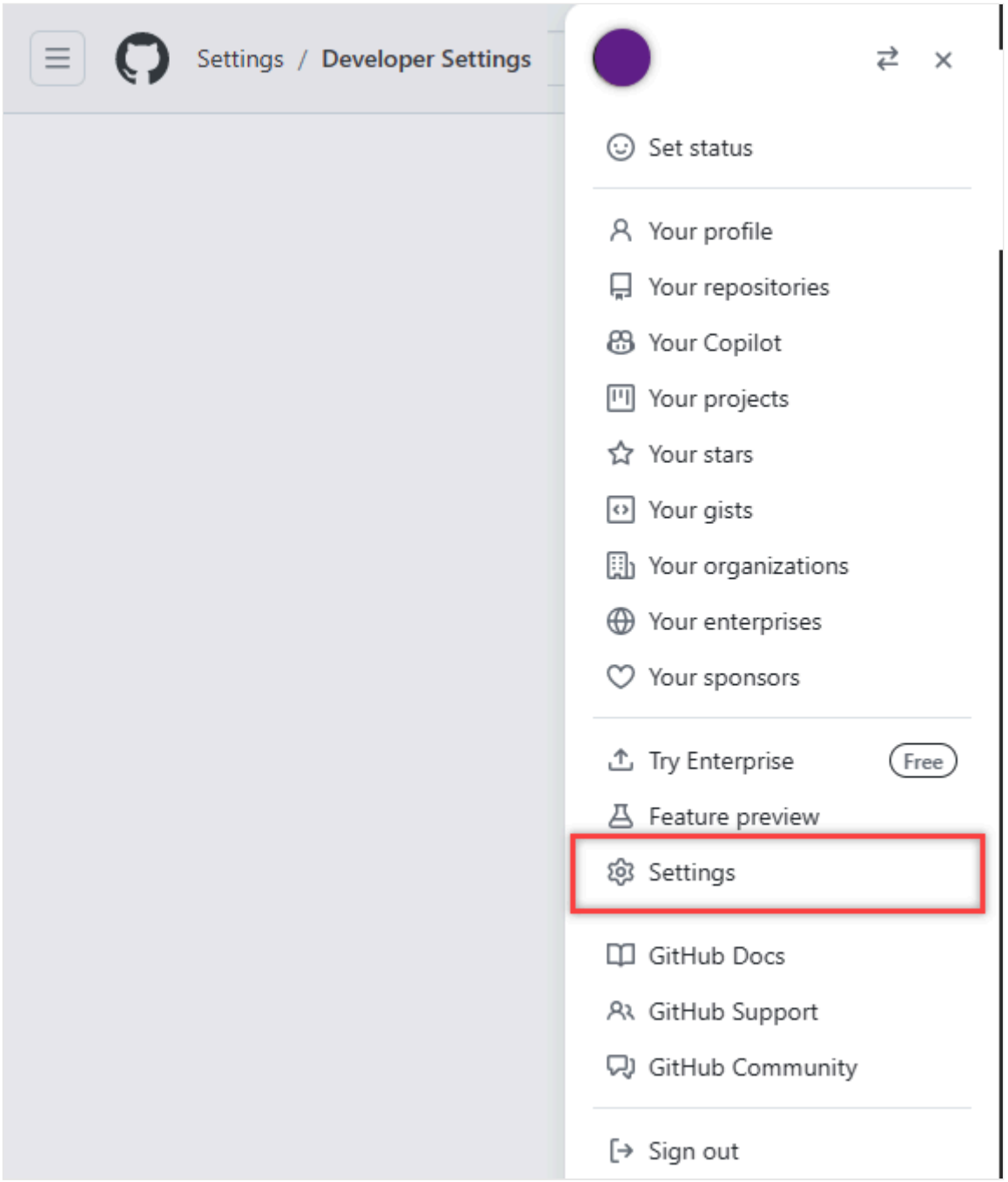
- OAuth
- PAT
- Username plus password

> **Note**
>
> With the Azure Boards app for GitHub, Azure Boards and Azure DevOps Services integrate with GitHub.com and GitHub Enterprise Server repositories. Azure DevOps Server 2019 and later versions support integration with GitHub Enterprise Server repositories only. Integration with other Git repositories is not supported.

Grant Azure Boards organization access

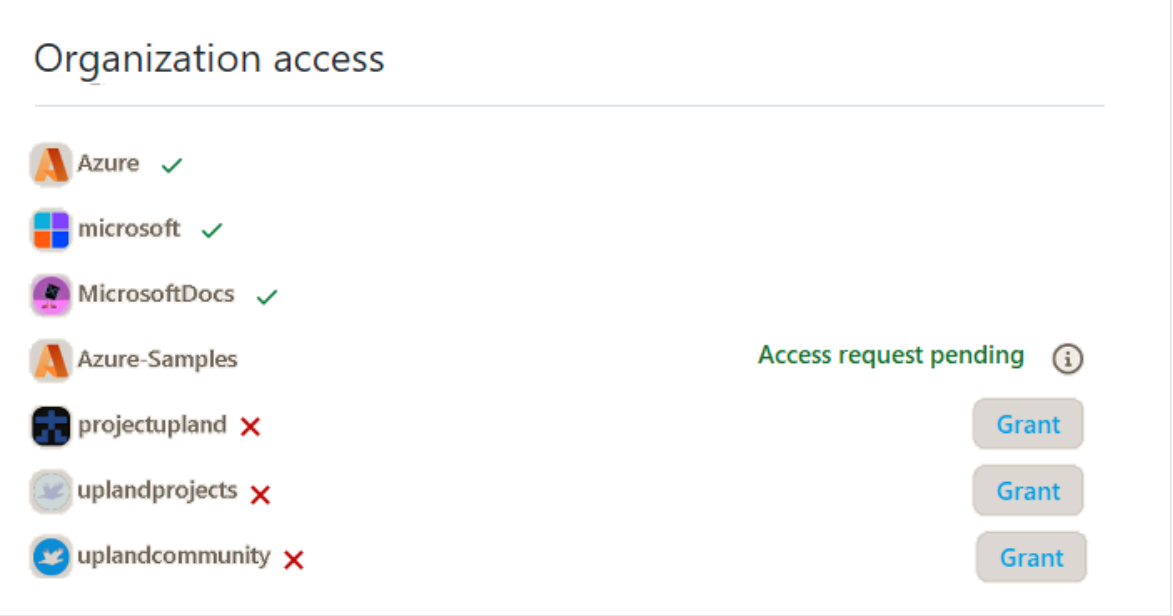
If the integration between Azure Boards and GitHub isn't working as expected, verify that you granted organization access.

1. From GitHub web portal, open **Settings** from your profile menu.



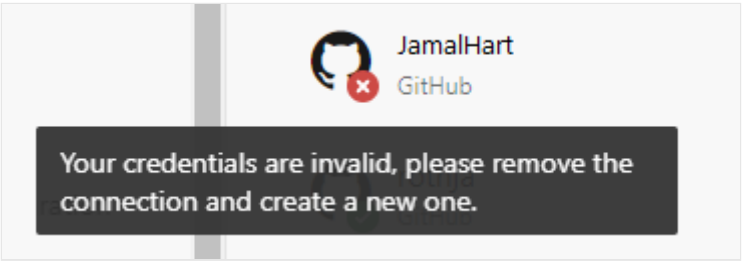
2. Select **Applications** under **Integrations** > **Authorized OAuth Apps** > **Azure Boards**.

3. Under **Organization access**, resolve any issues that might appear. Select **Grant** to grant access to any organizations that show as having an **Access request pending**.



Resolve access issues

When the Azure Boards connection to GitHub loses access, an alert status with a red-X appears in the user interface. Hover over the alert to see that the credentials are no longer valid. To fix the issue, remove the connection and create a new one.



To resolve this issue, consider the following items:

- **If the connection is using OAuth**:
  - The Azure Boards application had its access denied for one of the repositories.
  - GitHub might be unavailable/unreachable. This unavailability could be because of an outage in either service or an infrastructure/network issue on-premises. You can check service status from the following links:
    - GitHub
    - Azure DevOps

    Delete and recreate the connection to the GitHub repository. This recreated connection causes GitHub to prompt to reauthorize Azure Boards.
- **If the connection is using a PAT:**

- The PAT might be revoked or the required permission scopes changed and are insufficient.
- The user might not have admin permissions on the GitHub repo.

  Recreate the PAT and ensure the scope for the token includes the required permissions: `repo, read:user, user:email, admin:repo_hook`.

Resolve broken GitHub Enterprise Server connection

If you migrated from Azure DevOps Server to Azure DevOps Services with an existing GitHub Enterprise Server connection, your existing connection may not work as expected. Work item mentions within GitHub might be delayed or never appear in Azure DevOps Services because the callback URL associated with GitHub is no longer valid. Consider the following resolutions:

- **Remove and re-create the connection**: Remove and re-create the connection to the GitHub Enterprise Server repository. Follow the sequence of steps provided in Connect from Azure Boards documentation.
- **Fix the webhook URL**: Go to GitHub's repository settings page and edit the webhook URL, pointing to the migrated Azure DevOps organization URL: `https://dev.azure.com/{OrganizationName}/_apis/work/events?api-version=5.2-preview`

Connect to multiple Azure DevOps organizations

Connecting your GitHub repository to projects in multiple Azure DevOps organizations, such as dev.azure.com/Contoso and dev.azure.com/Fabrikam, can cause unexpected results with **AB#** mentions. Work item IDs aren't unique across Azure DevOps organizations, so **AB#12** could refer to a work item in either Contoso or Fabrikam. When a work item is mentioned in a commit message or pull request, both organizations might attempt to link to a work item with the matching ID, causing confusion.

To avoid this issue, connect a single GitHub repository to only one Azure DevOps organization.

> **Note**
>
> When you connect using the Azure Boards app for GitHub, the app prevents connections to multiple organizations. If a GitHub repository is connected to the wrong Azure DevOps organization, contact the owner of that organization to

remove the connection before adding the repository to the correct Azure DevOps organization.

Update XML definitions for select work item types

Update the XML definitions for the work item types if your organization uses the Hosted XML or on-premises XML process model to customize the work tracking experience and link to and view the GitHub link types from the Development section in the work item forms.

For example, to link user stories and bugs to GitHub commits and pull requests from the **Development** section, update the XML definitions for user stories and bugs.

Follow the sequence of tasks provided in Hosted XML process model to update the XML definitions. For each work item type, find the `Group Label="Development"` section, and add the following two lines in the code syntax to support the external link types: **GitHub Commit** and **GitHub Pull Request**.

```
            <ExternalLinkFilter Type="GitHub Pull Request" />
            <ExternalLinkFilter Type="GitHub Commit" />
```

After updating, the section should appear as follows.

```
<Group Label="Development">
   <Control Type="LinksControl" Name="Development">
      <LinksControlOptions ViewMode="Dynamic"
ZeroDataExperience="Development" ShowCallToAction="true">
         <ListViewOptions GroupLinks="false">
         </ListViewOptions>
         <LinkFilters>
            <ExternalLinkFilter Type="Build" />
            <ExternalLinkFilter Type="Integrated in build" />
            <ExternalLinkFilter Type="Pull Request" />
            <ExternalLinkFilter Type="Branch" />
            <ExternalLinkFilter Type="Fixed in Commit" />
            <ExternalLinkFilter Type="Fixed in Changeset" />
            <ExternalLinkFilter Type="Source Code File" />
            <ExternalLinkFilter Type="Found in build" />
            <ExternalLinkFilter Type="GitHub Pull Request" />
            <ExternalLinkFilter Type="GitHub Commit" />
         </LinkFilters>
      </LinksControlOptions>
   </Control>
</Group>
```

FAQs

Q: Some of my users in Azure DevOps have GitHub identities. Do I need to add them as new GitHub users to my organization?

A: No. Ask your users to sign out and then sign back in to the organization with their GitHub credentials from a fresh browser session. This action will establish their GitHub identities.

Q: I'm an organization Administrator and I enabled the policy for inviting GitHub users. Why can't I invite new GitHub users?

A: After changing the setting, sign out of Azure DevOps and then sign back in to the organization (`dev.azure.com/{organizationName}` or `organizationName.visualstudio.com`) with your GitHub credentials from a fresh browser session.

Q: I signed in with my GitHub credentials, but why can't I invite GitHub users?

A: Only organization or project Administrators can invite new users to join the organization. You might not have the required permissions to add new users. Work with your administrator to get the necessary permissions or ask them to add the user for you.

Next steps

Link GitHub commits and pull requests to work items

Related articles

- Install and configure the Azure Boards app for GitHub
- Configure status badges to add to GitHub README files
- Troubleshoot GitHub & Azure Boards integration
- Build GitHub repositories
- Change GitHub repository access

Feedback

Was this page helpful? Yes No

Provide product feedback

Set work tracking permissions

Article • 03/25/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

To manage work tracking effectively, you can grant or restrict access to different features. Do so by assigning specific permissions to users or groups for a particular object, project, or collection. You also can define custom rules for processes or projects that apply to specific users or groups, thereby controlling their actions accordingly. For most features, we recommend that you add users to the project's **Contributors** group, which grants comprehensive access and ensures a seamless and efficient work tracking experience.

> **Note**
>
> - For public projects, Stakeholder access gives users greater access to work tracking features and full access to Azure Pipelines. For more information, see Stakeholder access quick reference.

Prerequisites

- To set work tracking permissions, you must be a member of the **Project Administrators** group or have explicit permissions to manage the work tracking area as described in this article.

Understand roles and permission levels for work tracking

The following table summarizes the different permissions you can set at the object, project, or collection level. The team administrator role provides access to add and modify team resources. Also, see Default permissions for Boards, Backlogs, Sprints, Delivery Plans, Test Management, and Queries, further in this article.

Role or permission level

Functional areas set

Team administrator role

Add a team administrator

- Manage teams and configure team tools
- Define and edit team dashboards
- Add and manage team-level work item templates
- Add team administrators

Object-level permissions

- Modify work items under an area path
- Create and edit nodes under an area path or iteration path
- Define and edit queries or query folders
- Define and edit Delivery Plans

Project-level permissions

- Create work item tags
- Move work items out of a project
- Permanently delete work items
- Edit shared work item queries
- Add teams and team administrators)
- Edit project-level permissions

Project collection-level permissions

Includes all permissions you can set at the collection-level.

- Create, delete, or edit a process (Inheritance process model)
- Delete field from account (Inheritance process model)
- Manage process permissions (Inheritance process model)
- Edit collection level permissions
  Project collection-level permissions include all permissions you can set at the collection-level.

For more information, see Add groups.

Default permissions for Boards, backlogs, and sprints

Boards default permissions

Task

Readers

Contributors

Team admins
Project admins

View boards and open work items







Add work items to a board; update status through drag-and-drop





Reorder work items or reparent child items through drag-and-drop; update a field on a card





Add child items to a checklist





Assign to a sprint (from card field)





Configure board settings



Backlogs default permissions

Task

Readers

Contributors

Team admins
Project admins

View backlogs and open work items







Add work items to a backlog





Use bulk edit features





Add child items to a backlog item; prioritize or reorder a backlog; parent items using the Mapping pane; Assign items to a sprint using the Planning pane





Configure team settings, backlog levels, show bugs, work days off



Sprints default permissions

Task

Readers

Contributors

Team admins Project admins

View sprint backlogs, taskboards, and open work items







Add work items to a sprint backlog or taskboard





Prioritize/reorder a sprint backlog or taskboard; add child items to a backlog item; reassign items to a sprint using the Planning pane





View team capacity and work details





Set team capacity



Use bulk edit features





Define team sprints

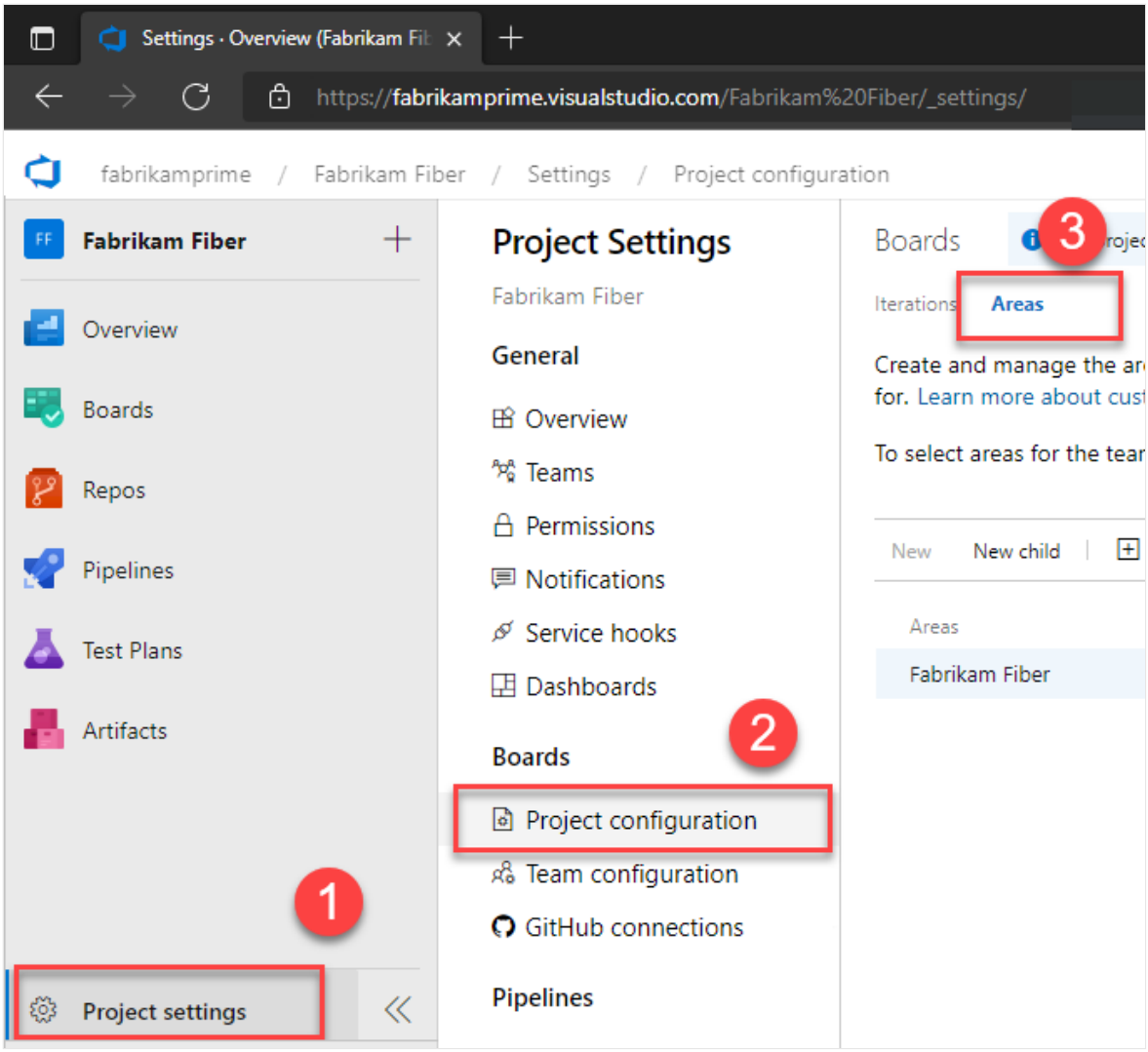
Create child nodes, modify work items under an area or iteration path

Area path permissions let you grant or restrict access to edit or modify work items, test cases, or test plans assigned to those areas. You can restrict access to users or groups. You can also set permissions for who can add or modify areas or iterations for the project.

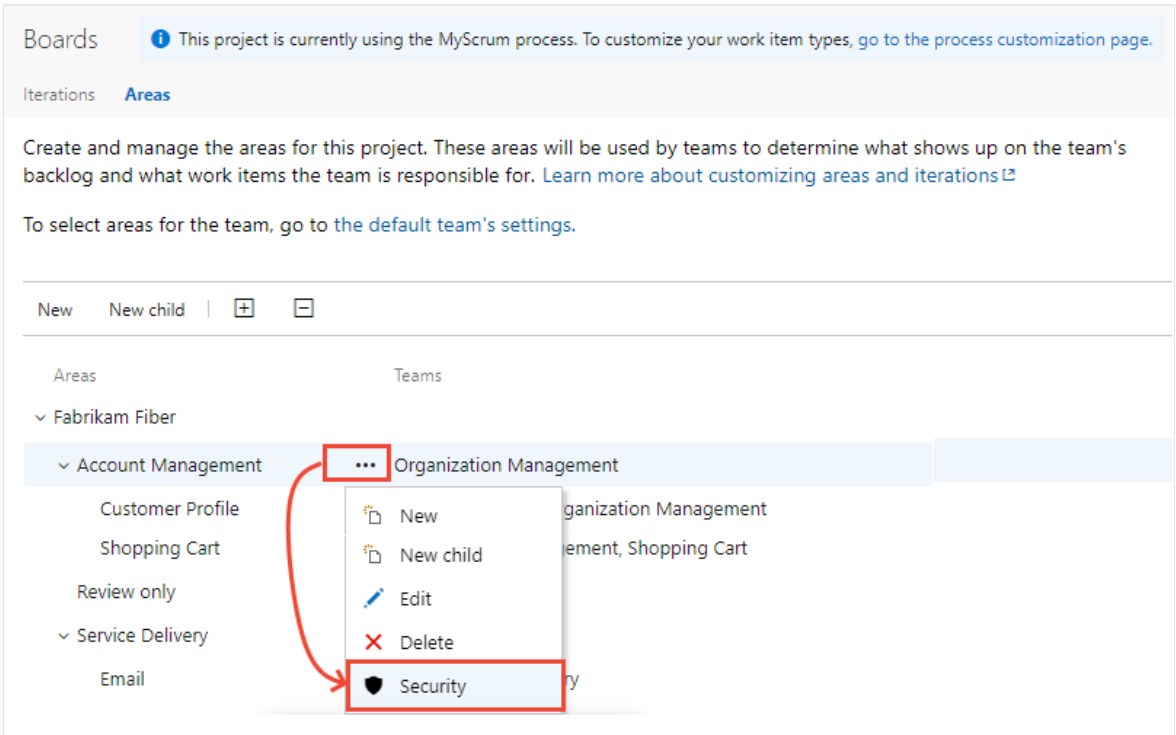
> **Note**
>
> Project members with permissions to create or edit **Area Paths** or **Iteration Paths** can't set team **Area Paths** and **Iteration Paths**. To configure team settings, you must be added to the team administrator role or be a member of the Project Administrators group.

Do the following steps to define both areas and iterations for a project.

1. Choose **Project settings** > **Project configuration** > **Boards**, and then select **Areas** or **Iterations** to modify Area Paths or Iteration Paths.

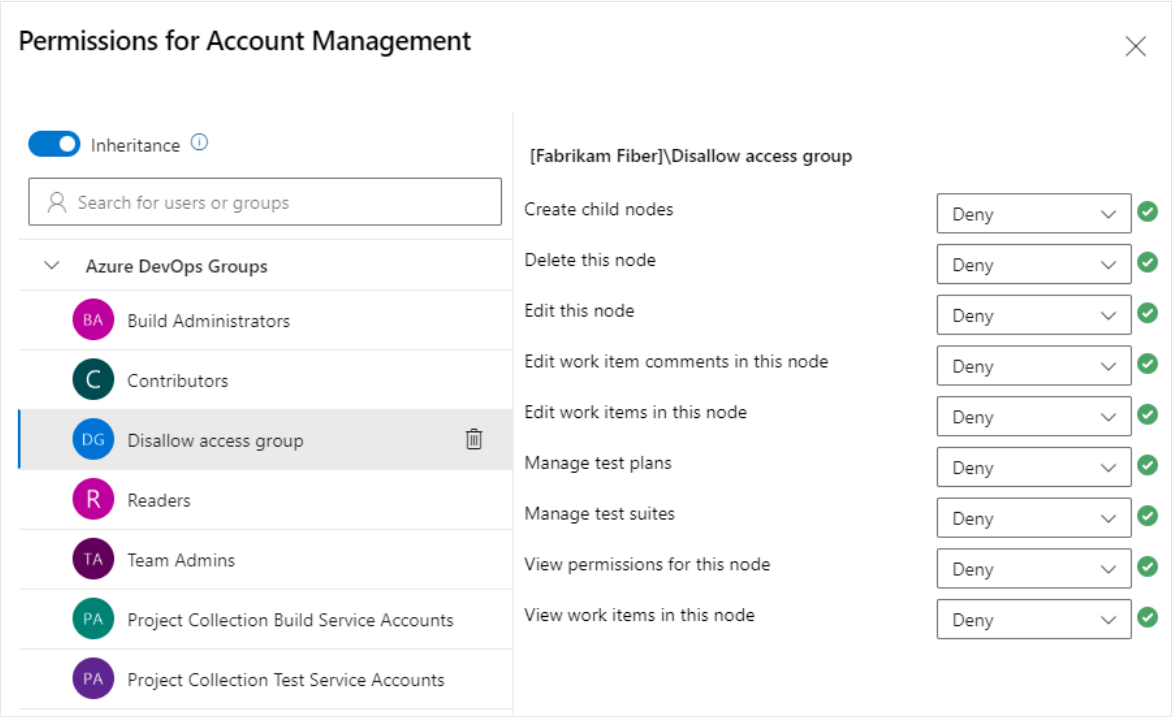


2. Choose the **...** context menu for the node you want to manage and select **Security**.



3. Select the group or project member, and then change the permission settings. To add a user or group, enter their name in the search box.

For example, here we added the *Disallow Access Group*, and disallowed members of this group the ability to view, modify, or edit work items in the **Account Management** area path.



You can specify two explicit authorization states for permissions: **Deny** and **Allow**. In addition, permissions can exist in one of the three other states. For more information, see About permissions, access, and security groups.

4. (Optional) Choose the **Inheritance** slider to disable inheritance. Disabling **Inheritance** persists all inherited permissions as explicit Access Control Entries (ACEs).

5. When you're done, close the dialog. Your changes automatically save.

Default permissions for work items

Task or permission	Readers	Contributors	Project admins
View work items in this node (Area Path permission)	✔		





Edit work items in this node (Area Path permission)





Edit work item comments in this node (Area Path permission)





Create tag definition





Change work item type (Project-level permission)





Move work items out of this project (Project-level permission)





Email work items







Apply a work item template





Delete and restore work items (Project-level permission) (able to restore from the Recycle bin)





Permanently delete work items (Project-level permission)



Provide feedback (through the Microsoft Feedback client)





Request feedback





> **Note**
>
> Work items are subject to rules applied to them. Conditional rules based on user or group membership are cached for your web browser. If you find yourself restricted to update a work item, you may have encountered one of these rules. If you believe you've encountered an issue that doesn't apply to you, see Work item form IndexDB caching issues. For more information, see Rules and rule evaluation.

Use custom rules

Custom rules don't control permissions, but they affect whether a user can modify a work item or set the value of a work item field. Azure Boards supports the following work tracking customizations that support business workflows.

Expand table

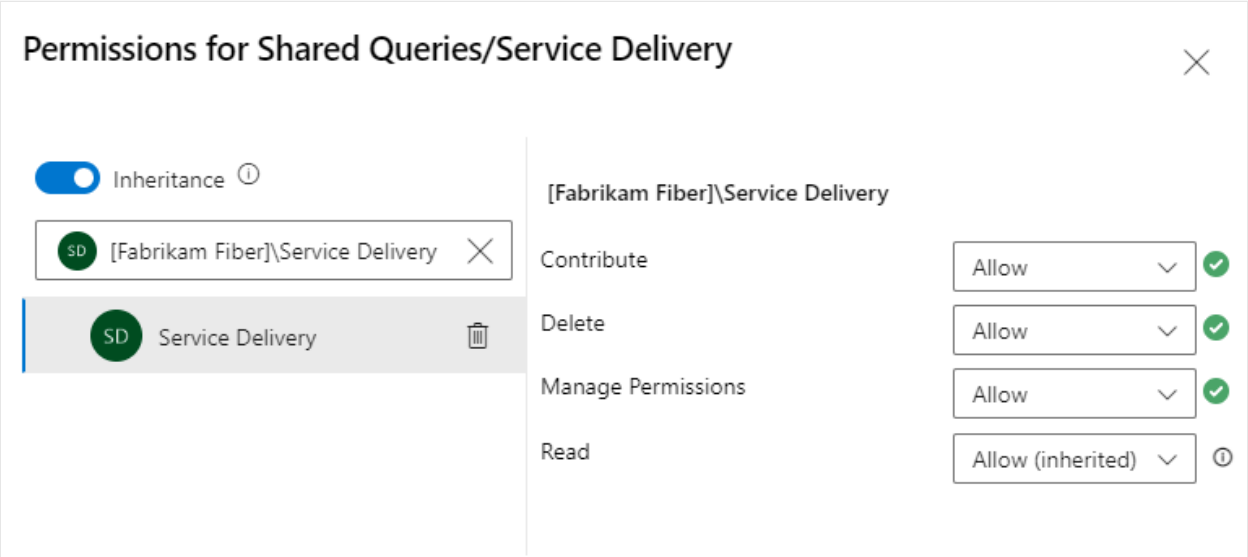
Customization	Examples
Apply rules upon work item creation, state change, and specified state.	- Make a field read-only - Make a field required
Apply rules when a field value is empty, set to a specific value, or change or not changed to a value.	- Clear the value of a field if it's empty or meets certain criteria - Set a predefined value for the field if it's empty or meets specific conditions - Copy the value of one field to another field - Hide a field based on certain conditions or values
Apply rules that dictate what state a work item can get moved to from a given state.	- Reassign a work item based on state changes - Specify that a work item can only transition from "State A" to "State B" - Manage the state transitions of parent work items based on the state changes of their child work items
Apply rules based on user or group membership of the user modifying a work item.	Specify rules that restrict a group from creating a work item, transitioning a work item to a closed or completed state, or changing the value of a field

There are some restrictions for applying custom rules to system fields. For example, you can't specify rules that set or clear the value for **Area Path** or **Iteration Path** as they're system fields. For more information, see Rules and rule evaluation and Sample rule scenarios.

Set permissions on queries or query folders

You can specify who can add or edit query folders or queries at the object-level. To manage permissions for a query or query folder, you must be the creator of the query or folder, a member of the Project Administrators or Project Collection Administrators group or granted explicit access through the object's **Security** dialog.

Query folder permissions dialog



For more information, see Create managed queries to list, update, or chart work items.

Default permissions for queries

> **Tip**
>
> By default, Contributors can't create and save shared queries. We recommend that Project Administrators create a query folder for each team and give the team administrators or the team group query permissions to manage their folder. You need **Delete** permissions to rename or move a shared query or folder, and **Contribute** permissions for the folder where you move the query to. To learn more, see Set permissions on queries and query folders.

Task	Readers	Contributors	Project admins
View and run managed queries, view query charts	✔	✔	✔
Create and save managed **My queries**, query charts			





Create, delete, and save **Shared queries**, charts, folders



Adhoc searches are powered by a semantic search engine.

Set permissions for work item tags

By default, all users of the Contributors group can create and add tags to work items. To set permissions for a group or user to restrict this ability, you can set the **Create tag definition** to **Deny** at the project-level. To learn how, see Change the permission level for a project-level group.

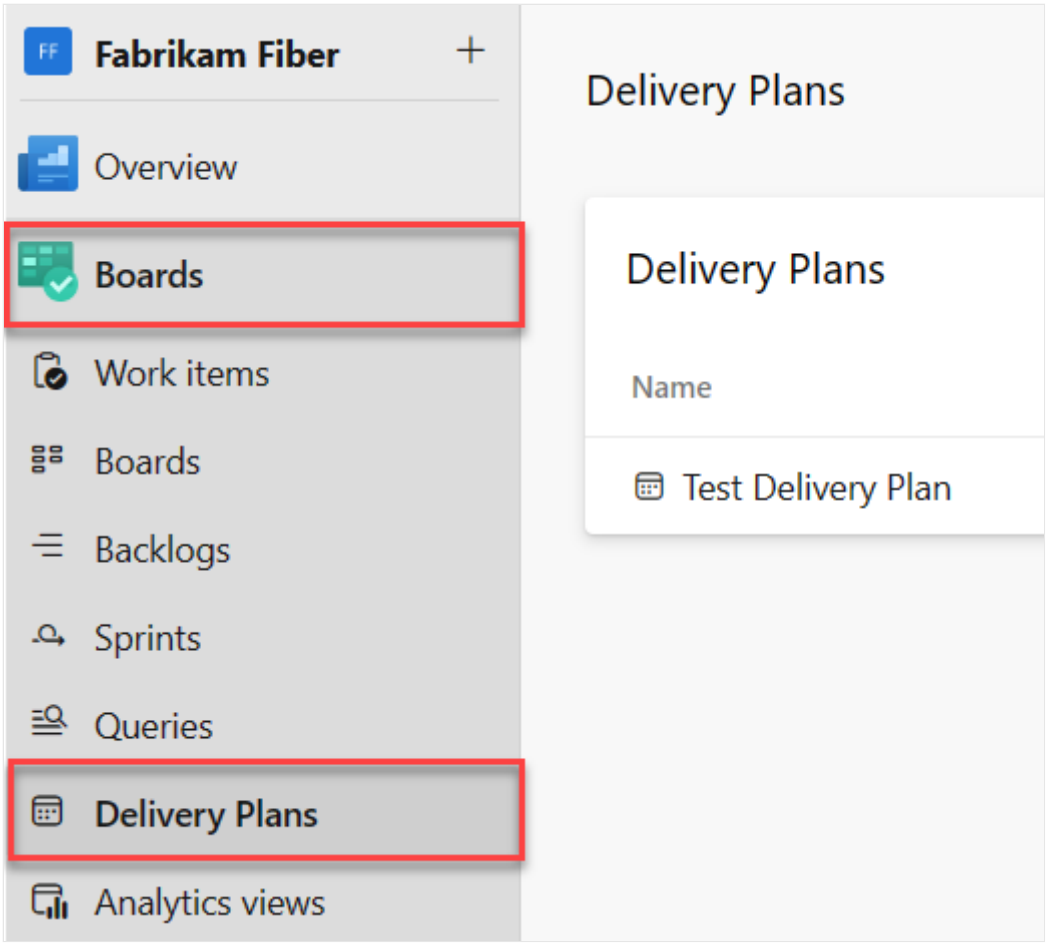
Manage permissions for Delivery Plans

Delivery Plans are an object within a project. You can manage permissions for each plan like the way you manage permissions for shared queries or query folders. The creator of a Delivery Plan and all members of the Project Collection Administrators and Project Administrators groups have permissions to edit, manage, and delete plans.

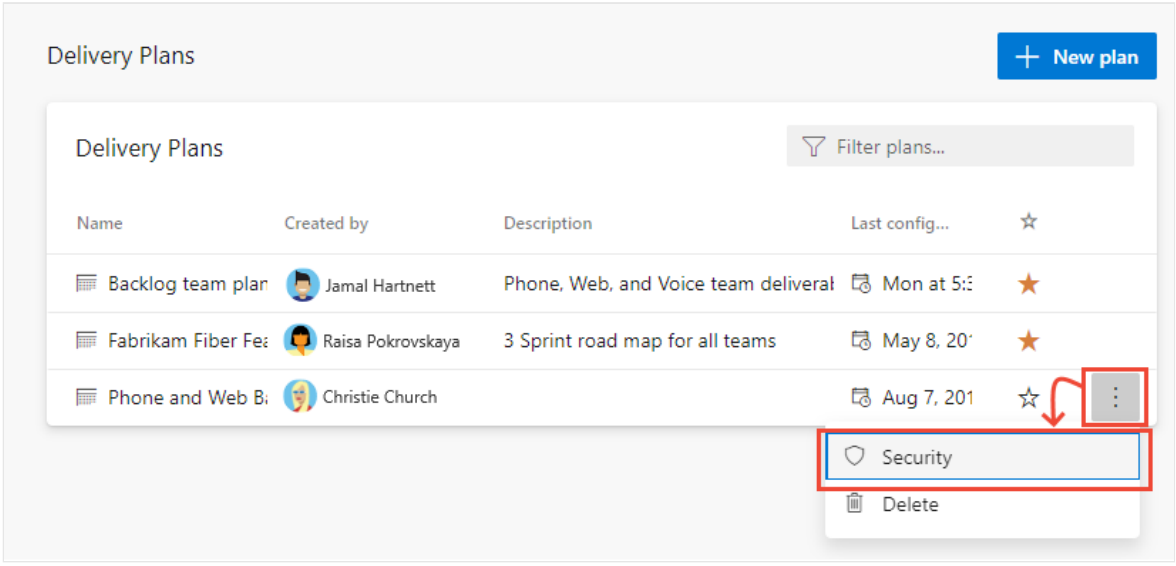
Users granted **Stakeholder** access for private projects have no access to delivery plans, while users granted **Stakeholder** access for public projects has the same access as regular Contributors granted **Basic** access. For a comparison chart of Stakeholder versus basic access, see the Feature Matrix.

To edit the permissions for a Delivery Plan, you must be the creator of the plan, a member of the Project Administrators or Project Collection Administrators group, or granted explicit permission through the plan's **Security** dialog.

1. Open **Boards** > **Delivery Plans**.

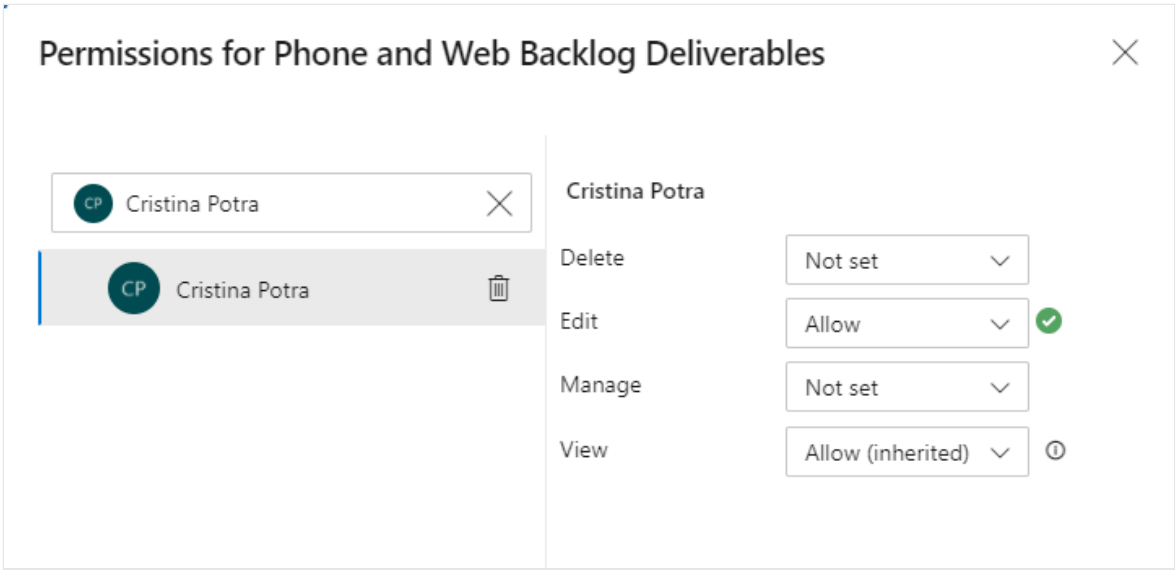


2. To grant permissions to a group or user to manage or edit a specific plan, choose ⋮ **More options** to open the **Security** dialog for the plan.



3. Add a user, team group, or other security group who you want to grant permissions to or restrict access. For details, see Change project-level permissions. By default, nonadministrators can't delete or edit a plan.

4. With the user or group selected, set the permission you want them to have to **Allow**. **Manage** set to **Allow** enables the user to manage permissions for the plan.



5. When you're done, close the dialog. Your changes automatically save.

Default permissions for Delivery Plans

Task

Readers

Contributors

Team admins
Project admins

View delivery plans

✔

✔

✔

Create, edit, or delete a delivery plan, Contributors can only edit or delete plans that they create

✔

✔

Manage permissions for a delivery plan, Contributors can only manage permissions for plans that they create

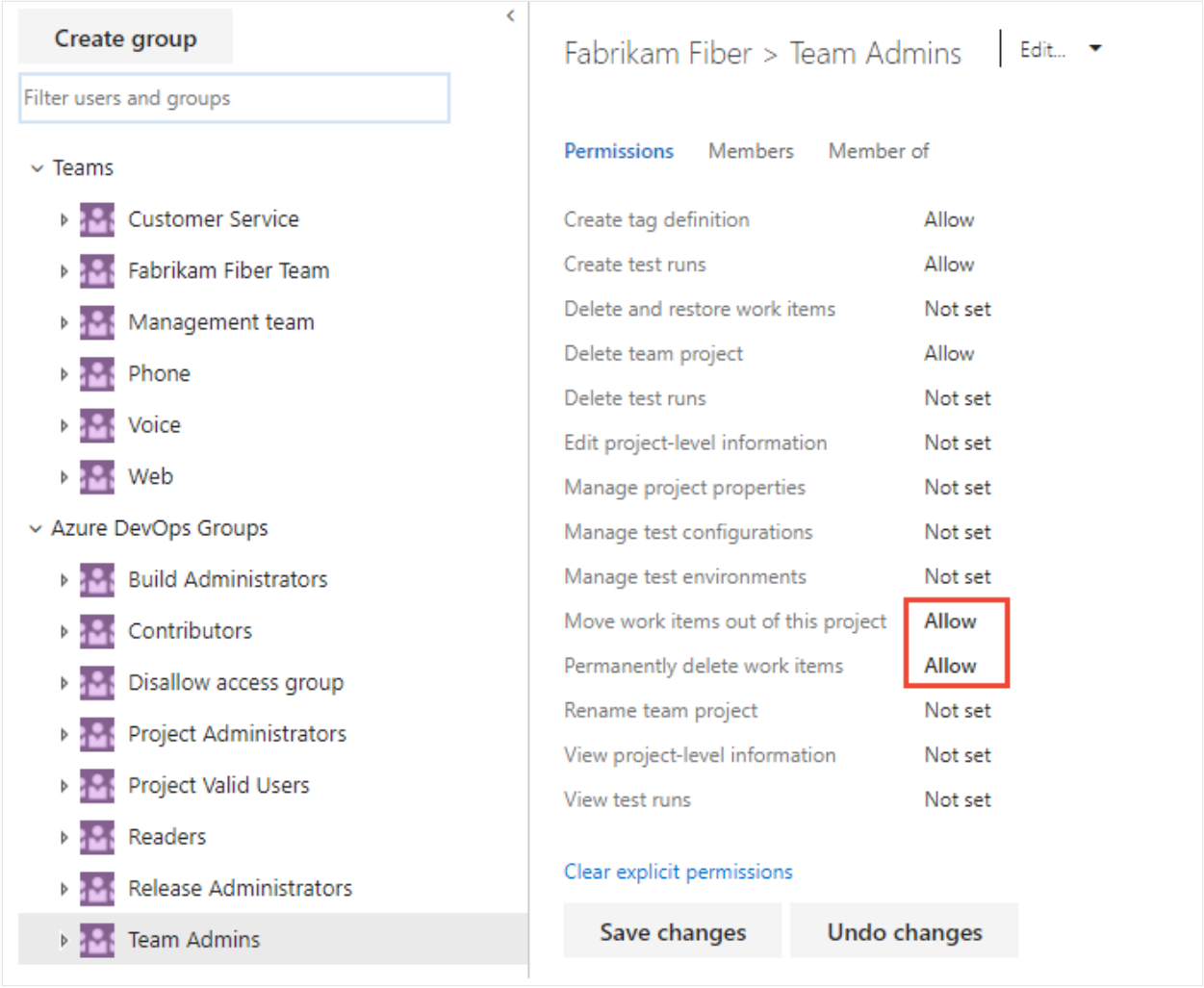
Move or permanently delete work items

By default, Project Administrators and Contributors can change the work item type and delete work items by moving them to the **Recycle Bin**. Only Project Administrators can permanently delete work items and test artifacts. Project admins can grant permissions to other team members as needed.

For example, as a project admin you can grant a user, team group, or other group you've created to have these permissions. Open the Security page for the project and choose the user or group you want to grant permissions. To learn how to access project-level **Security**, see Change project-level permissions.

> **Note**
>
> The **Move work items out of this project** permission requires the Inherited process model for the project.

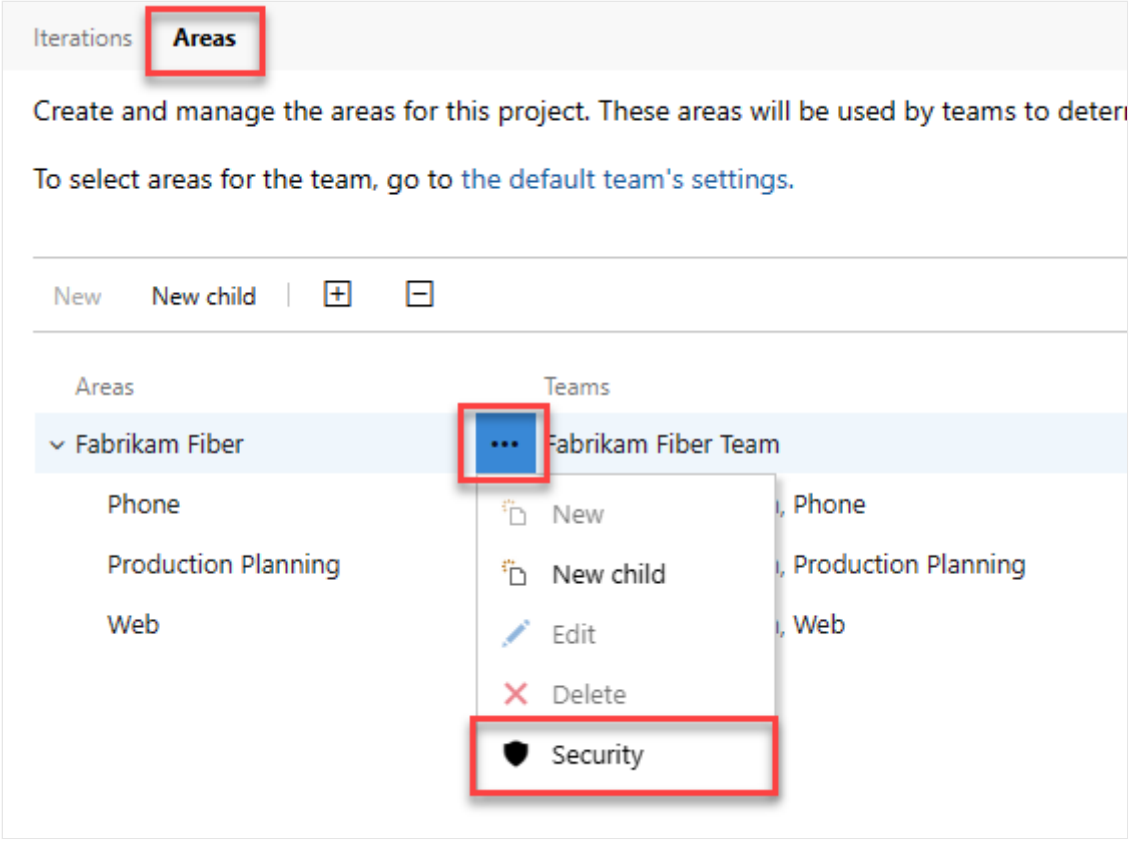
In the following example, we granted members who are assigned to the team administrator role, and who belong to the Team Admin group, permissions to move work items to another project and permanently delete work items.



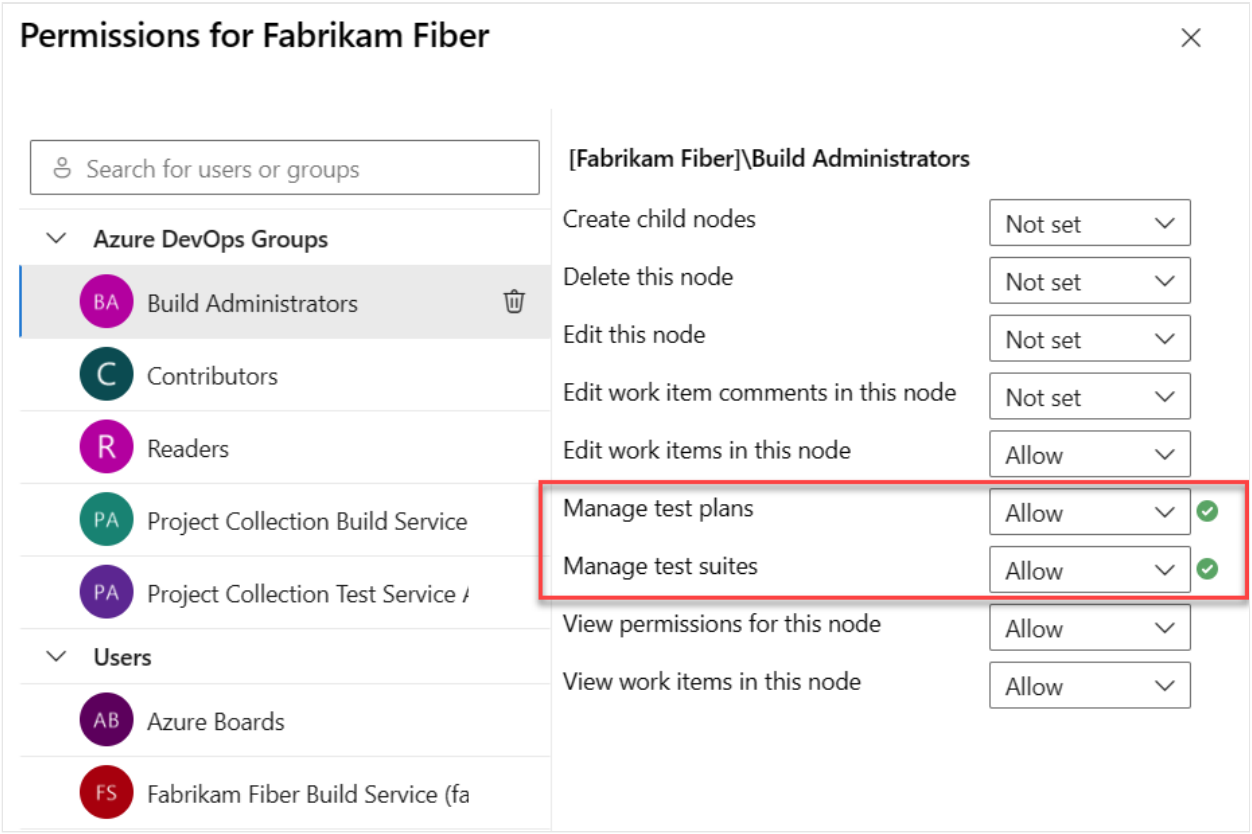
Manage test plans and test suites

In addition to the project-level permissions set in the previous section, team members need permissions to manage test artifacts that are set for an area path.

Open the **Security** page for area paths and choose the user or group you want to grant permissions.



Set the permissions for **Manage test plans** and **Manage test suites** to **Allow**.



To have full access to the Test feature set, your access level must be set to Basic + Test Plans. Users with Basic access and with permissions to permanently delete work items and manage test artifacts can only delete orphaned test cases.

Default permissions for test management

Test plans, test suites, test cases and other test artifacts are specific work item types that support manual and exploratory testing. For more information, see Set test permissions at the project level.

Permission	Level	Readers	Contributors	Project Admins
View test runs	Project-level	✔	✔	✔
Create test runs Delete test runs	Project-level		✔	✔
Manage test configurations Manage test environments	Project-level		✔	✔

Create tag definition
Delete and restore work items

Project-level





Permanently delete work items

Project-level



View work items in this node

Area Path







Edit work items in this node
Manage test plans
Manage test suites

Area Path





> ⓘ **Note**
>
> The **Change work item type** permission doesn't apply to test-specific work items. Even if you choose this feature from the work item form, changing the work item type is disallowed.

Area permissions for web-based test case management and test execution control access to the following actions.

The **Manage test suites** permission enables users to do the following tasks:

- Create and modify test suites
- Add or remove test cases to/from test suites
- Change test configurations associated with test suites
- Modify the suite hierarchy by moving a test suite

The **Manage test plans** permission enables users to do the following tasks:

- Create and modify test plans
- Add or remove test suites to or from test plans
- Change test plan properties such as build and test settings

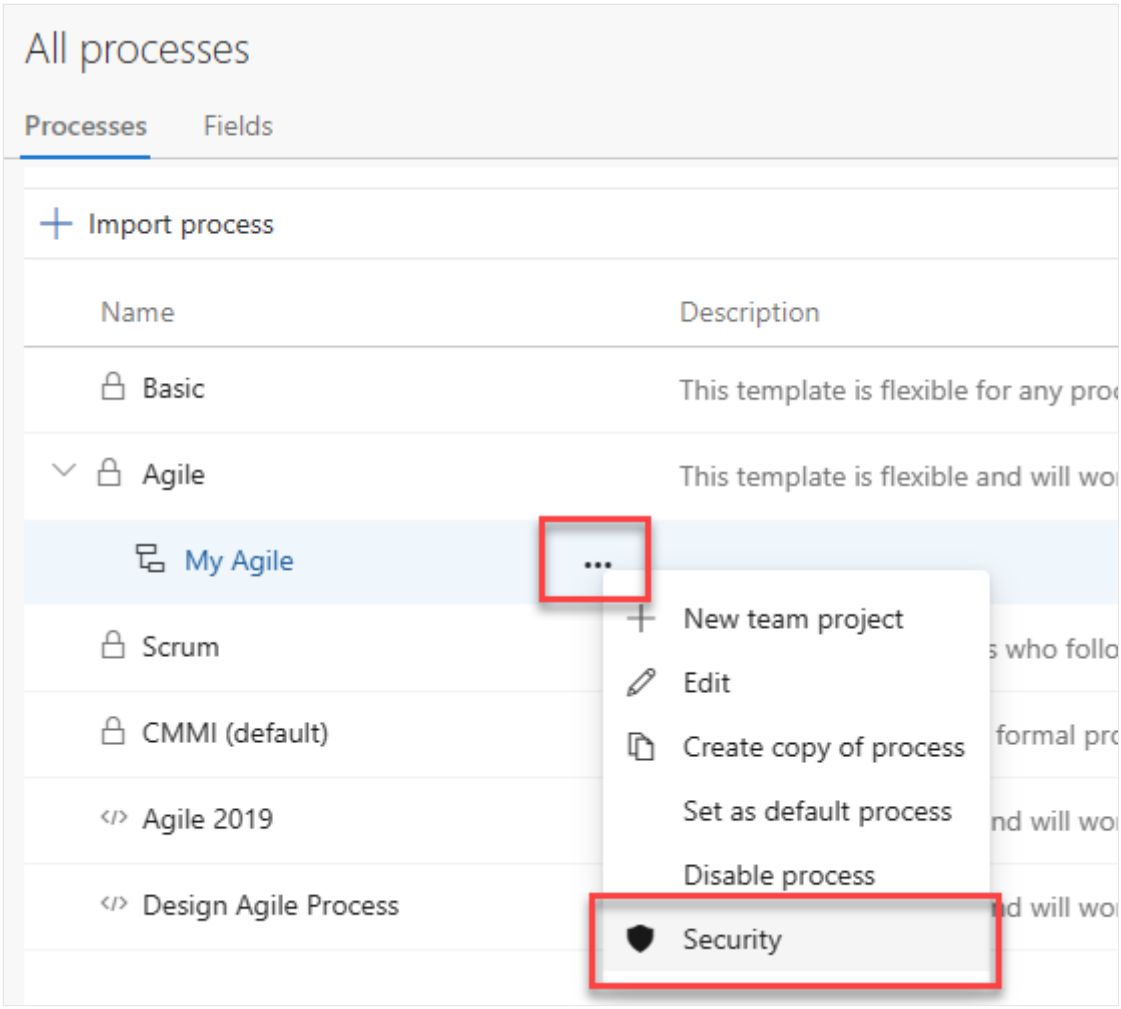
Customize an inherited process

By default, only Project Collection Administrators can create and edit processes. However, these admins can grant permissions to other team members by explicitly setting the **Create process**, **Delete process**, or **Edit process** permissions at the collection level for a specific user.

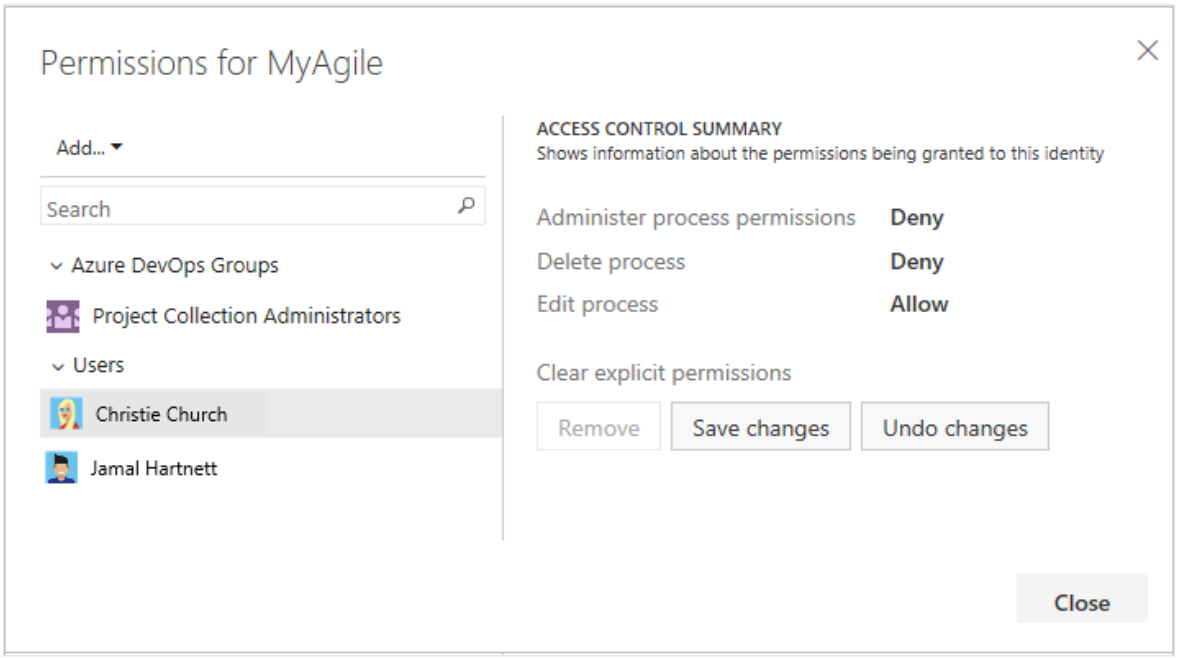
To customize a process, you need to grant **Edit process** permissions to a user account for the specific process.

> ⓘ **Note**
>
> Users added to the **Project-Scoped Users** group can't access Process settings if the **Limit user visibility and collaboration to specific projects** preview feature is enabled for the organization. For more information including important security-related callouts, see Manage your organization, Limit user visibility for projects and more.

1. Open the ... context menu for the inherited process and choose **Security**. To open this page, see Customize a project using an inherited process.



2. Enter the user name, set the applicable permissions to **Allow**, and then exit. The page automatically saves.



Note

Each process is a securable unit and has individual access control lists (ACLs) that govern creating, editing, and deleting inherited processes. At the collection level, project collection administrators can choose the inherited processes. When you create a new inherited process, the process creator and project collection administrators have full control of the process and can also set individual ACLs for other users and groups to edit and delete the process.

More access options for work items

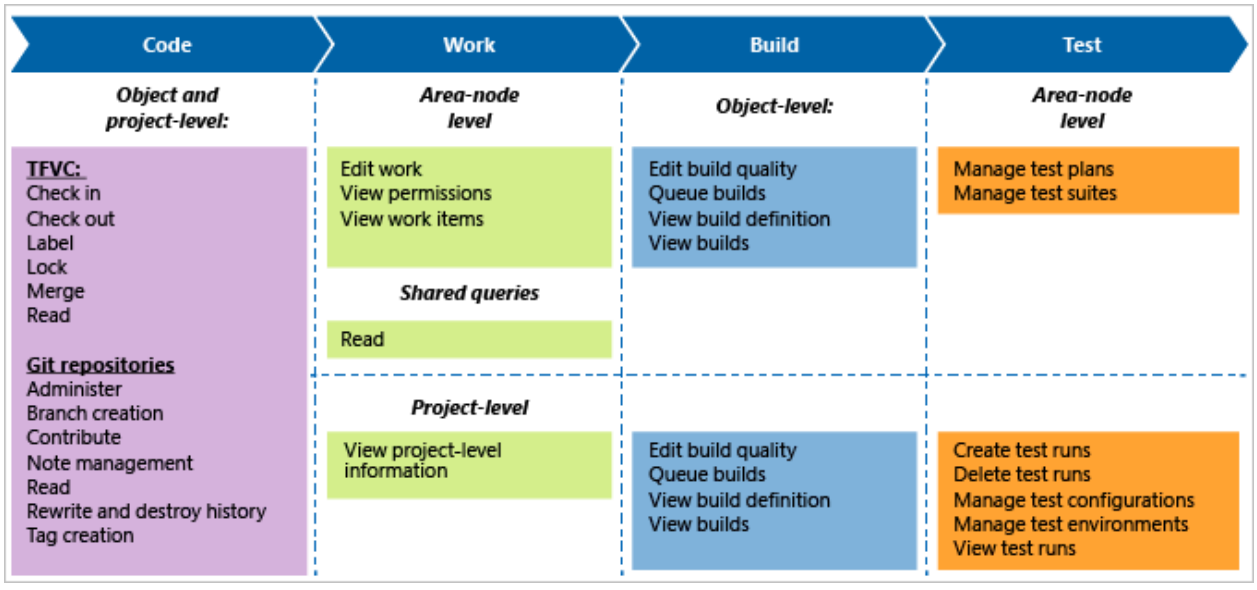
For more information about options for customizing work item types to support restrictions, see Restrict access, Restrict modification of work items based on a user or group.

Grant team members additional permissions

For teams to work autonomously, you may want to provide them with permissions that they don't have by default. Suggested tasks include providing team administrators or team leads permissions to:

- Create and edit child nodes under their default area path
- Create and edit child nodes under an existing iteration node
- Create shared queries and folders under the Shared Queries folder.

By default, team members inherit the permissions afforded to members of the project Contributors group. Members of this group can add and modify source code, create and delete test runs, and create and modify work items. They can collaborate on a Git project or collaborate with other team members and check in work to the team's code base (TFVC).



Related articles

- Grant or restrict access
- Rules and rule evaluation
- Permissions and groups reference
- Delete test artifacts

Feedback

Was this page helpful? Yes No

Provide product feedback

Manage and configure team tools

Article • 11/13/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

If you're a team administrator, you have the flexibility to tailor your backlogs and boards to align with your team's workflow. In case you require a new team, you can ask a Project Administrator group member to create one for you, which takes only a minute. Team administrators have the ability to set up and oversee all team tools.

Team administrators perform the following tasks for team tools:

- Add team members
- Add another team administrator
- Configure areas and iteration paths
- Configure backlogs, boards, and general settings
- Configure and manage team dashboards
- Configure team notifications

Prerequisites

Permissions:

- To manage teams, be a team administrator or be a member of the Project Administrators group.
- To add a team, be a member of the Project Administrators group. For more information, see Add teams.

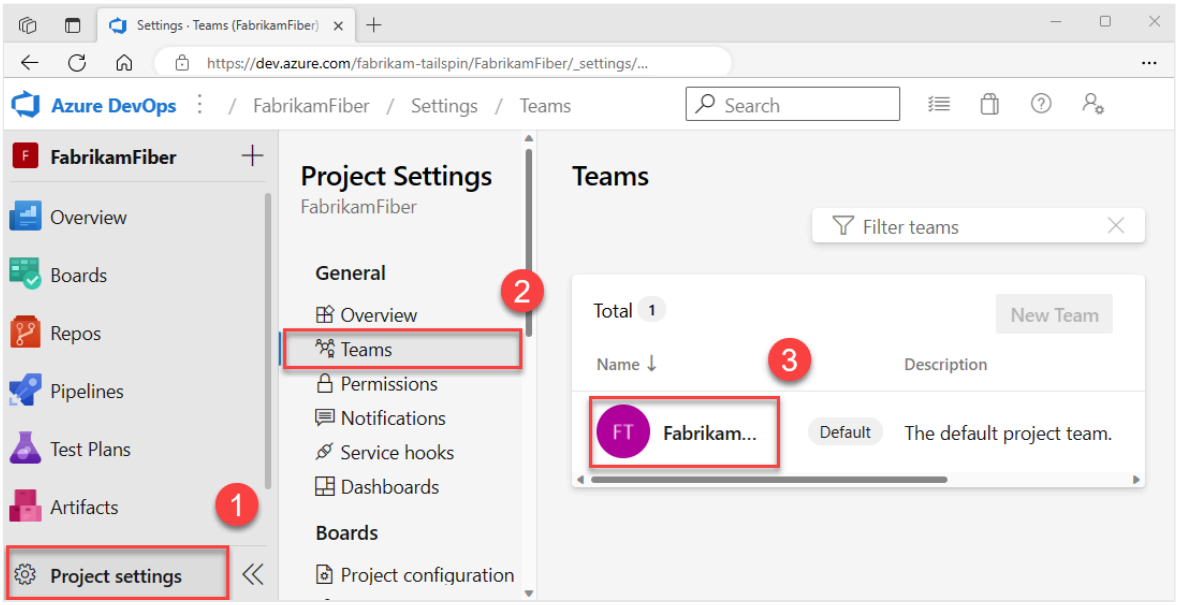
> **Note**
>
> For more information, see the following articles:
>
> - Use backlogs to manage projects
> - Configure and customize Azure Boards
> - Create a project using the process of your choice
> - Customize your work tracking experience
> - Create and manage inherited processes

Open your team profile

Open your team profile to quickly access items defined for your team.

1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`), and then open your project.

2. Select **Project settings** > **Teams** > your team name.

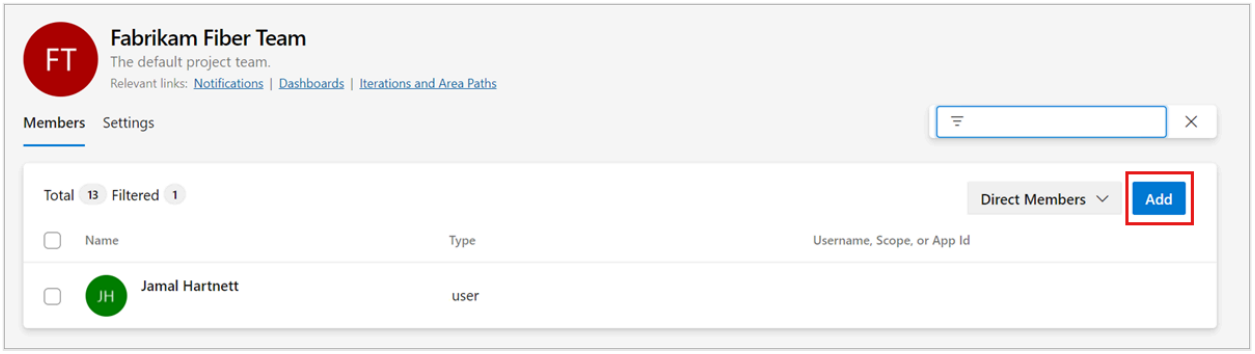


::: moniker-end

Add users to a team

Tools like capacity planning, team alerts, and dashboard widgets operate within the scope of a team. They automatically access the user information of team members to facilitate planning tasks or issue alerts.

To add users to a team, see Add users to a project or specific team.



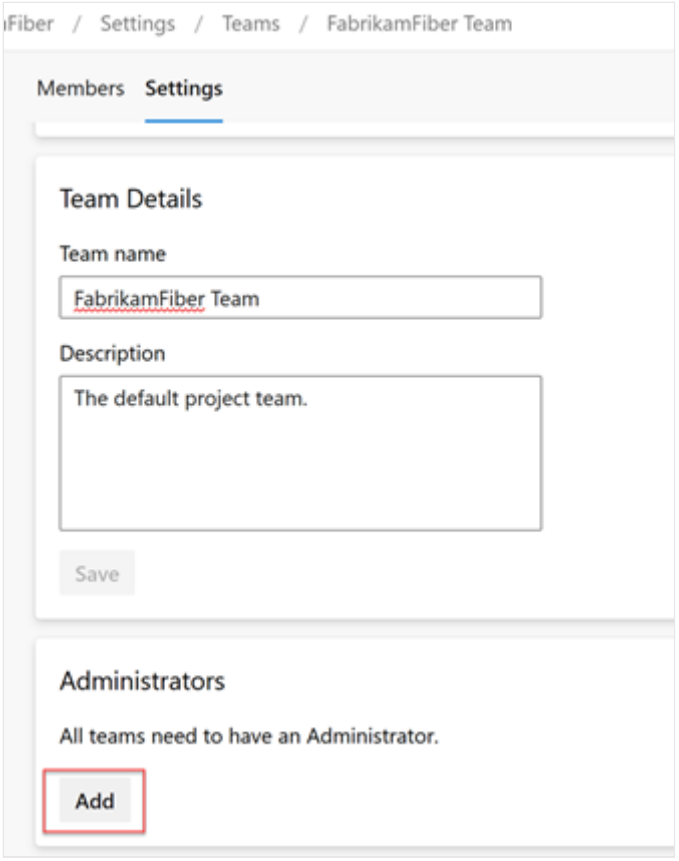
All members of a team can favorite team artifacts and define work item templates. For more information, see:

- Set personal or team favorites
- Use templates to add and update work items.

If team members don't have access to all the features they want, make sure they have the permissions needed for those features.

Add an administrator

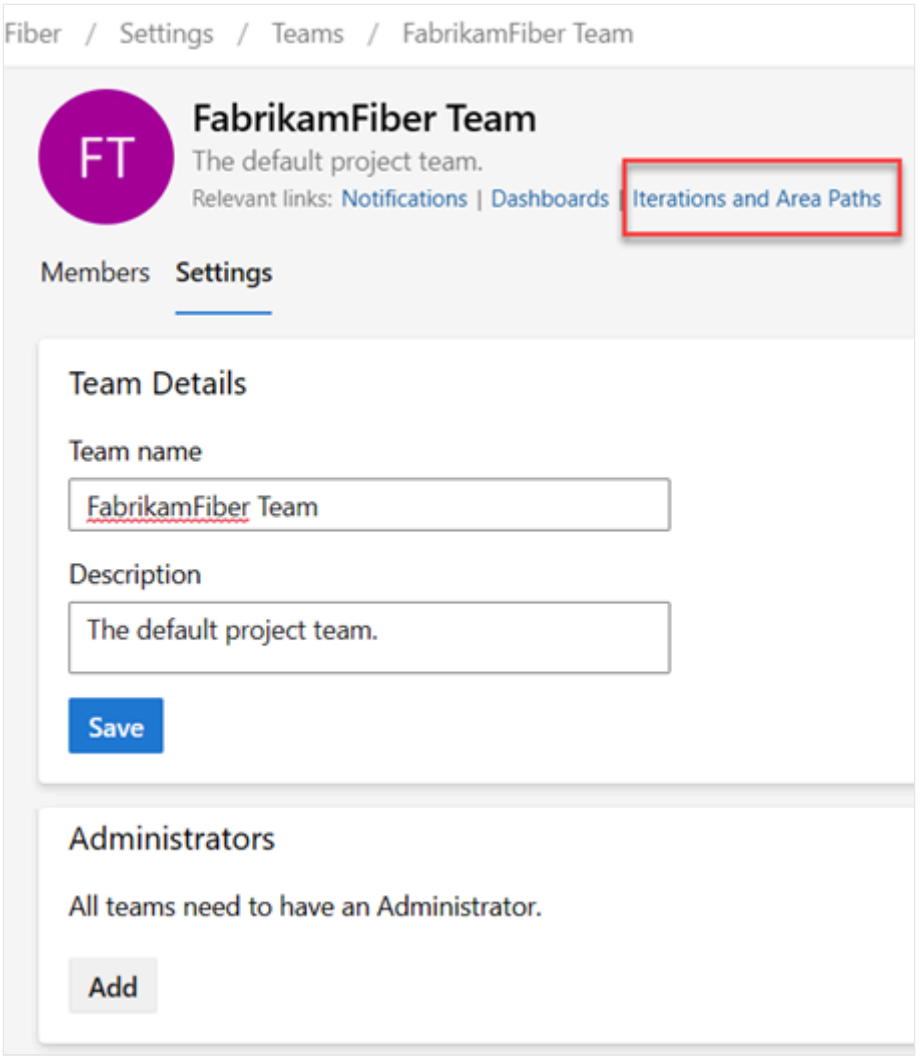
When you add a team to a project, a Project Administrator should add one or more team administrators.



Configure team areas and iterations

Many Agile tools rely on the team's configured area and iteration paths. For more information, see About teams and Agile tools.

After project administrators add the project's area and iteration paths using Set area paths and Set iteration paths, team administrators can choose the relevant area and iteration paths for their team. These settings influence a wide range of Agile tools that the team can access.



Settings include making the following associations for each team:

- **Select team area paths**
  Can select the default area paths associated with the team. These settings affect many Agile tools available to the team.
- **Select team iteration paths or sprints** Can select the default area paths associated with the team. These settings affect many Agile tools available to the team.

For more information, see Define area paths and assign to a team and Define iteration paths and configure team iterations.

Configure team backlogs, boards, and general settings

As a team administrator, you have the flexibility to customize your team's workflow to suit your needs. One way to do so is by choosing which backlog levels are active for your team. For instance, a feature team might only want to display the product backlog, while a management team might prefer to show the feature and epic backlogs only.

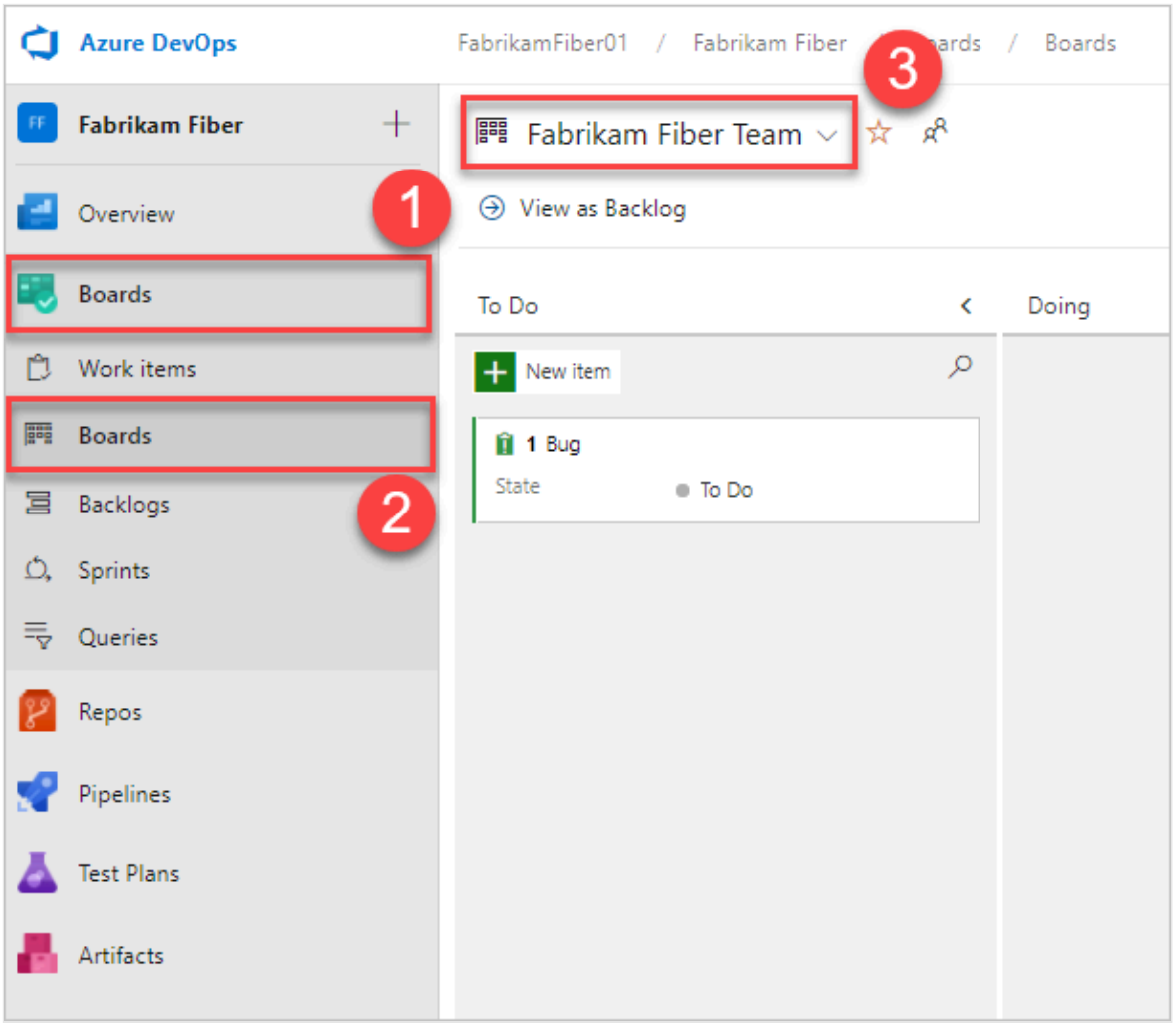
Also, you can choose how to treat bugs within your workflow, either as user stories and requirements or as tasks.

Another way to customize your team's workflow is by selecting nonworking days for the team. By doing so, sprint planning and tracking tools can automatically take these days off into account when calculating capacity and sprint burndown.

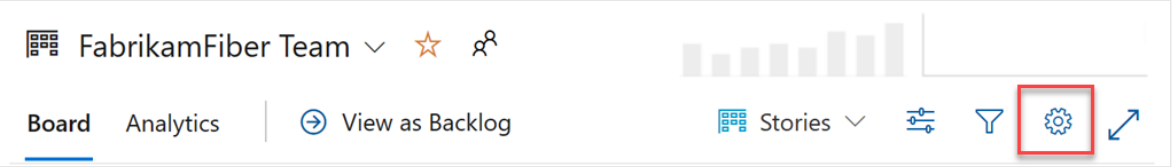
Most of these team settings can be easily configured from the common configuration dialog, providing a convenient way to manage your team's workflow in one central location. You can also set team automation rules to update work items when child item states change.

> ⓘ **Note**
>
> For more information, see Backlogs, boards, and plans. In case you don't see the desired work items on your backlog or board, see Set up your backlogs and boards.

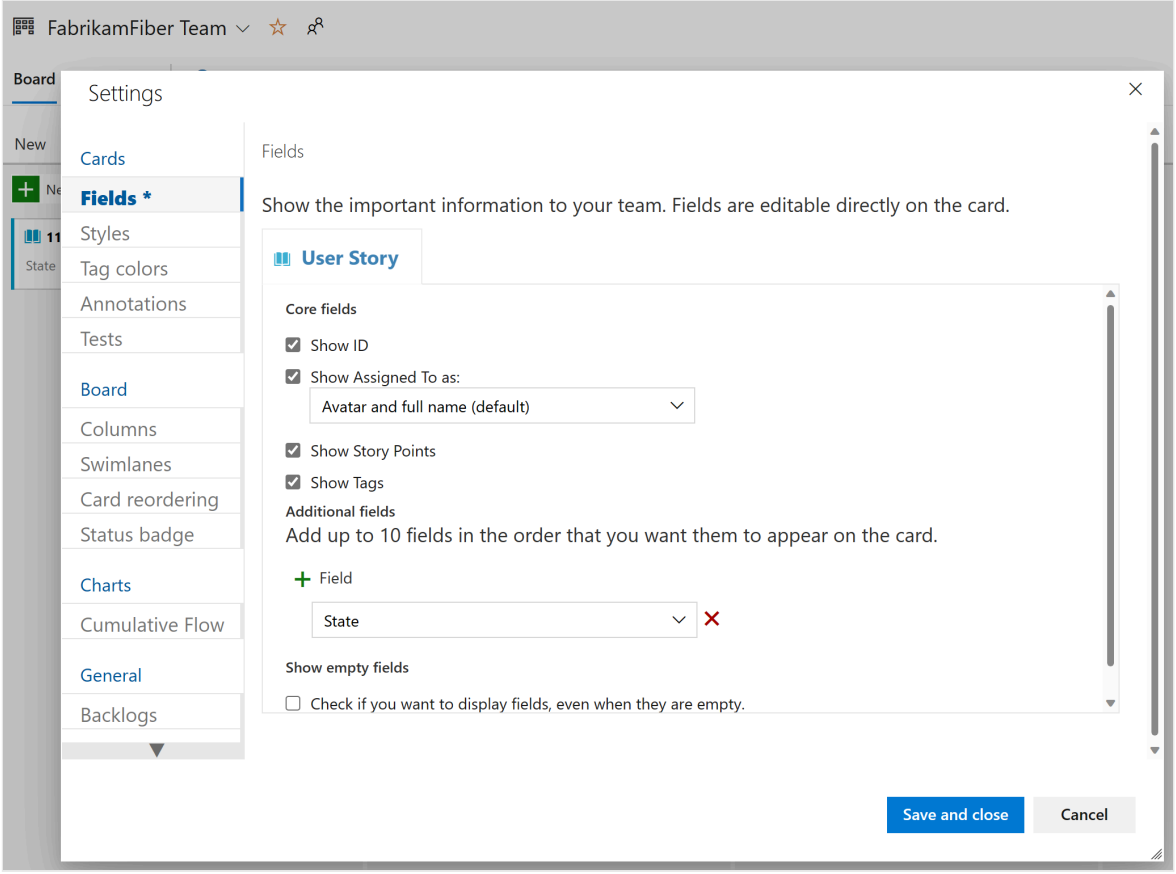
1. Check that you selected the correct project, and then choose **Boards** > **Boards**, and select the correct team from the team selector dropdown menu. For more information, see Use breadcrumbs and selectors to navigate and open artifacts.



2. Choose **Team settings** to configure the board and set general team settings.



3. Choose a tab under any of the sections—**Cards, Board**, **Charts**, and **General**—to configure the cards or boards, the cumulative flow chart, or other team settings. When you're done configuring the settings, select **Save and close**.



Team administrators have complete control over customizing their team's boards for both the product and portfolio backlogs. To set up a board, you can define the columns and work-in-progress (WIP) limits through the common configuration dialog. For more information, see Board overview and Board quickstart.

For detailed information on each configuration option, you can explore the following articles:

General

- Backlogs
- Working with bugs

Cards

- Add fields
- Define styles
- Add tag colors
- Enable annotations
- Configure inline tests

Boards

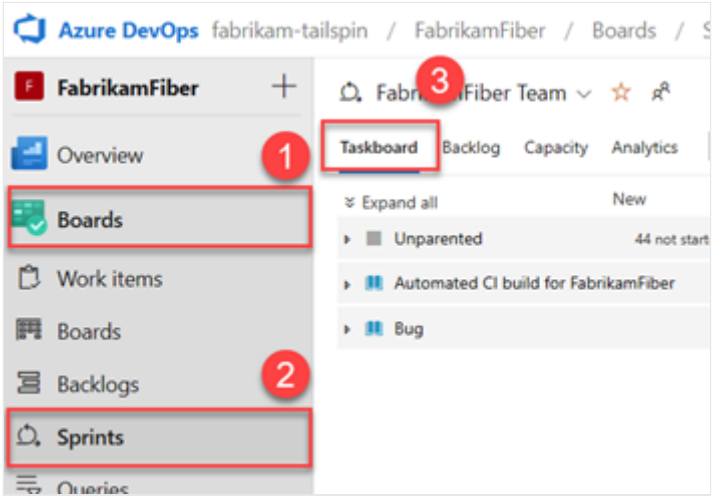
- Add columns
- Split columns
- WIP limits
- Definition of Done
- Add swimlanes
- Card reordering
- Configure status badges

Chart

- Configure cumulative flow chart

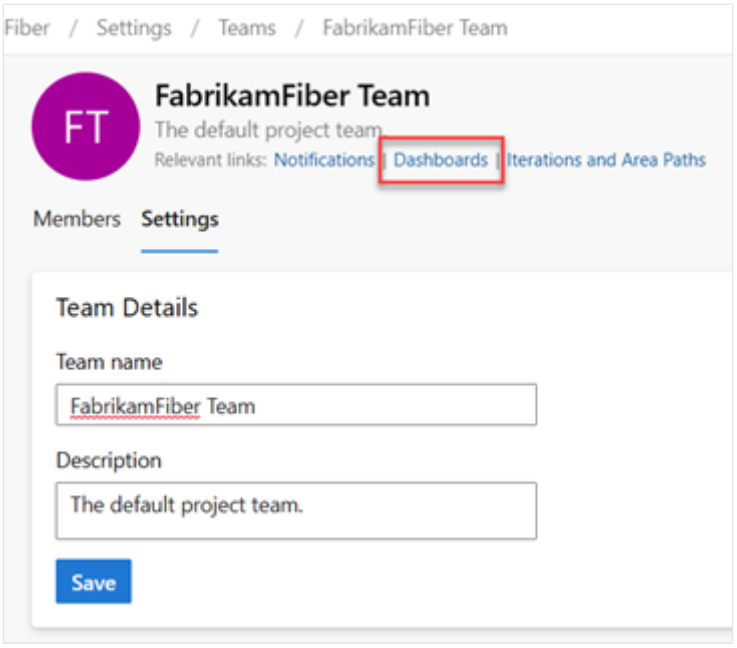
Configure sprint taskboards

Similar to boards, you can customize each sprint Taskboard to support information-rich color-coded cards and columns. For more information, see Customize sprint taskboards.



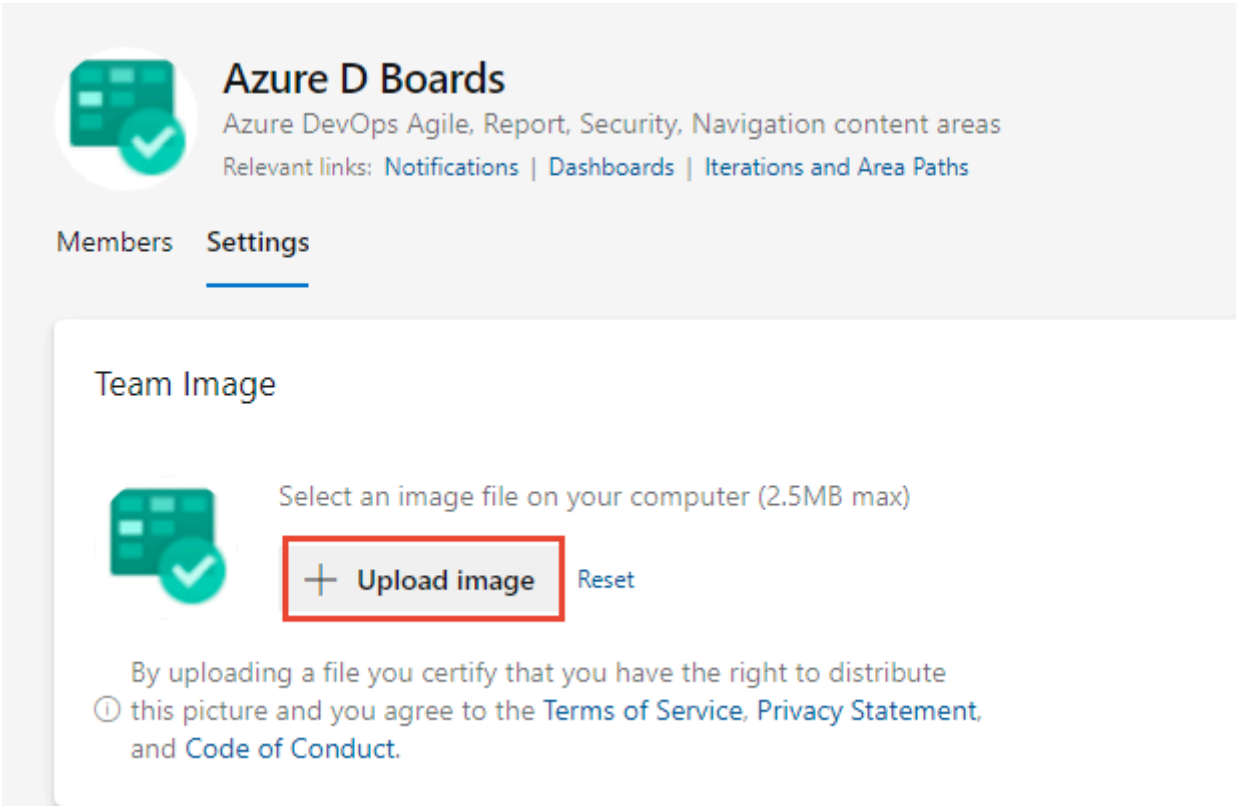
Add and manage team dashboards

By default, all team members can add and edit team dashboards. In addition, team administrators can manage permissions for team dashboards. For more information, see Add and manage dashboards.



Update team name, description, and image

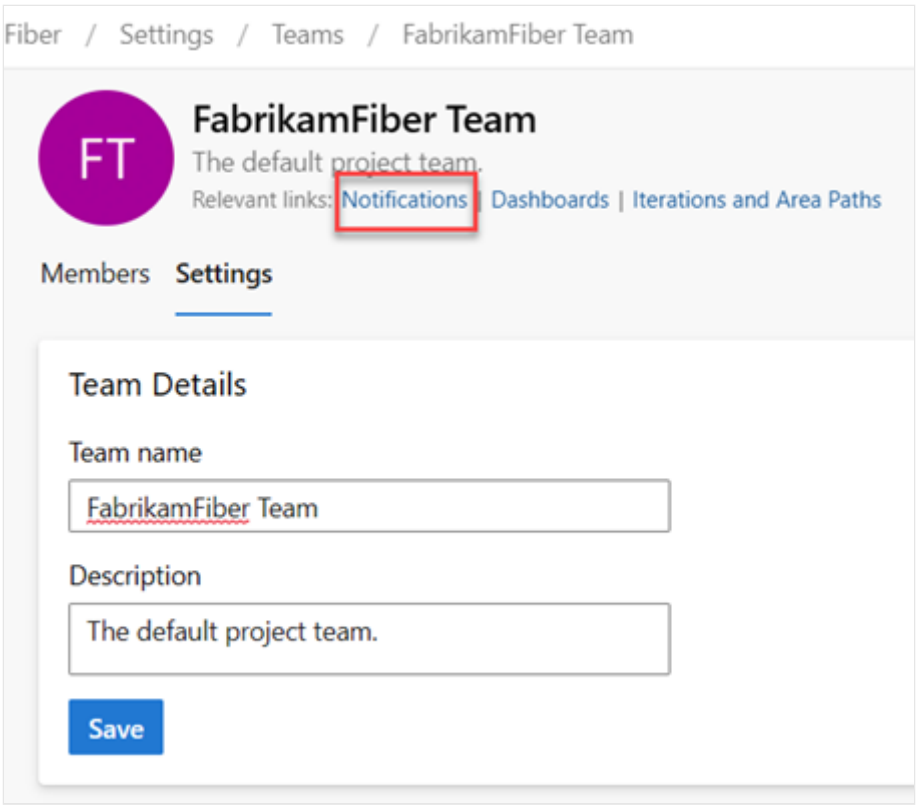
Team settings also include the team name, description, and team profile image. To add a team picture, select the image icon. The maximum file size is 2.5 MB and 2560 x 1,024 px, and then we resize to 200 x 200.



Manage notifications

Team administrators have the ability to add and edit alerts, allowing the team to receive email notifications as changes occur to work items, code reviews, source control files,

and builds. Various alerts are predefined for each team. For more information, see Manage team alerts.



Related articles

- About projects and scaling your organization
- About teams and Agile tools
- Add teams
- Add a team administrator
- Automate work item state transitions

Feedback

Was this page helpful? Yes No

Provide product feedback

Select backlog navigation levels

Article • 11/01/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Each team has the flexibility to choose and configure multiple backlog levels according to their needs. Feature teams might focus on their product backlog, while management teams might display both feature and epic backlogs (the default portfolio backlogs). You can add and configure these backlog levels through team settings to suit your team's workflow.

If you need more portfolio backlogs, see the following articles based on the process model you use:

- **Inheritance**: Customize your backlogs or boards for a process
- **On-premises XML**: Add portfolio backlogs.

For an overview of process models, see Customize your work tracking experience.

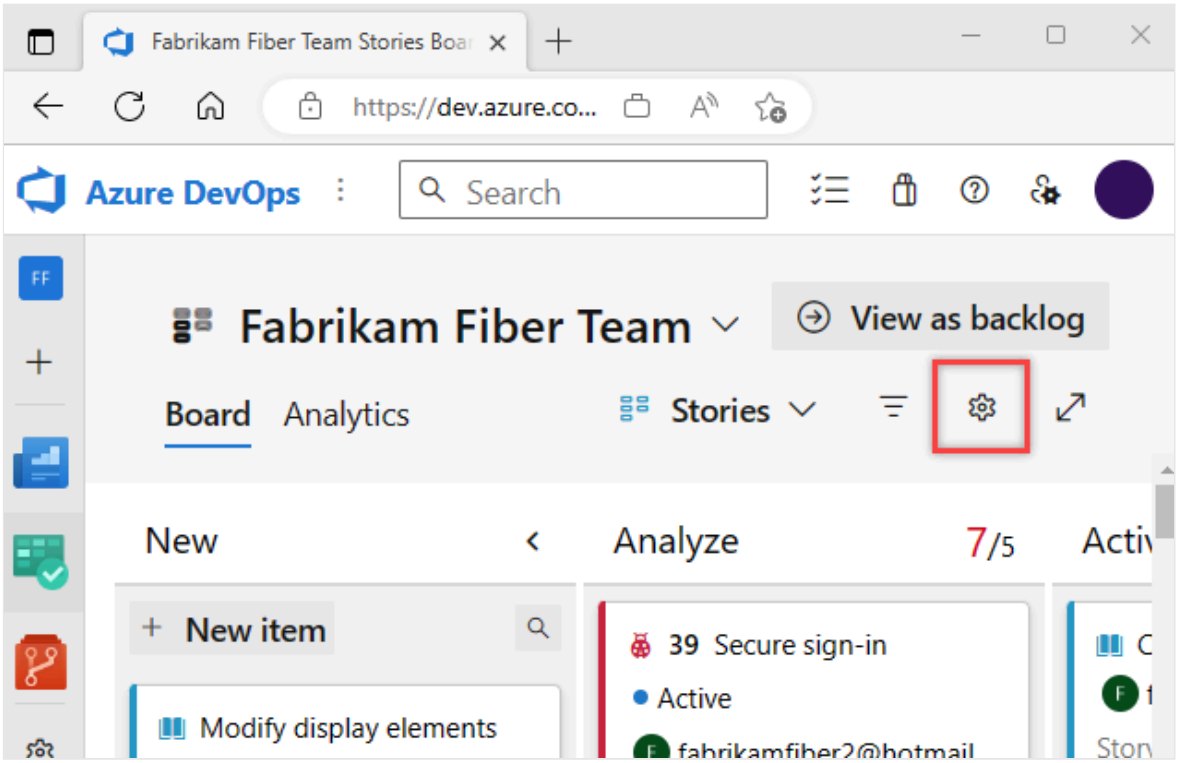
Prerequisites

Permissions: Be the Team Administrator or a member of the **Project Administrators** group.

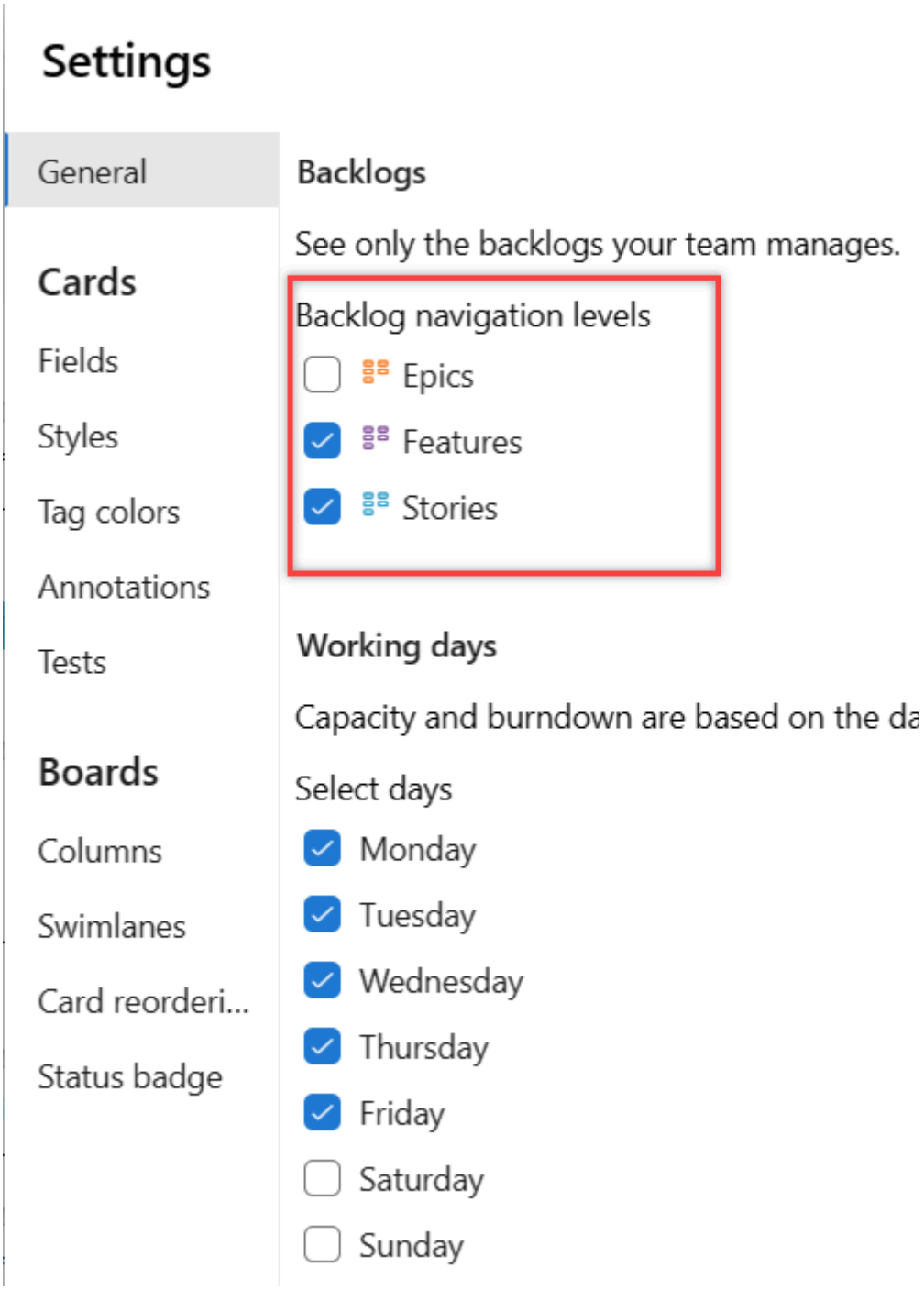
Set your team's preferences for backlog levels

This setting affects the backlog and board views for all team members, ensuring a consistent experience across the team. You can modify this setting from either the backlog or board view, depending on your preference. In this article, we show how to adjust the setting from the board view.

1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`).
2. Select **Boards**, and then open your board.
3. Select **Configure team settings**.



4. Check the boxes of those backlog levels you want your team to manage.



5. Select **Save**.

6. To see the changes, open or refresh your team's backlog.

::: moniker-end Your customized backlog and board view reflect the selected backlog levels.

Related articles

- Get started with Agile tools to plan and track work
- Understand backlogs, boards, and plans
- Create your backlog
- Define features and epics

- Organize your backlog

Feedback

Was this page helpful? Yes No

Provide product feedback

Show bugs on backlogs and boards

Article • 11/12/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

As your team identifies code defects or bugs, they can add them to the backlog and track them similar to tracking requirements. You can schedule bugs to be fixed within a sprint along with other tasks.

When you track bugs as *requirements*, they appear on the product Backlogs and boards. When you track bugs as *tasks*, the bugs appear on Sprint Backlogs and Taskboards. For more information about other work item types, see Add other work item types to backlogs or boards.

You can define your team's tracking setting for the Agile, Scrum, and Capability Maturity Model Integration (CMMI) processes. The Bug work item type isn't defined for the Basic process, so there isn't a team setting for Basic. Instead, you should track bugs and code defects using the Issue work item type.

> **Note**
>
> *Requirements* specify expectations of users for a software product. In Azure Boards, requirements are defined by work items that appear on your product backlog. Based on the process selected for your project, requirements correspond to User Story (Agile), Product backlog item (Scrum), Issue (Basic), or Requirement (CMMI) work item types. They also belong to the Requirements category, which manages the work item types that appear on the product backlog.

Permissions: Be a team administrator or a member of the Project Administrators group.

Choose from options for bug tracking

The following table summarizes the options teams have for tracking bugs. Before you make your choice, we recommend you review the information provided in Define, capture, triage, and manage bugs, which provides an overview of the Bug work item type and supported tools for managing bugs.

Option

Choose when you want to...

Track bugs as Requirements

- Prioritize, or stack rank, bugs along with requirements
- Estimate Bug effort for forecasting
- Update bug status on board
- Include Bugs in Velocity charts and Cumulative Flow Diagrams
- Be able to use the Forecast tool to support sprint planning
- Drag bugs to the Planning pane to assign bugs to a sprint
- View Bugs on Delivery Plans

> ⓘ **Note**
>
> - Bugs are assigned to the Requirements Category

Track bugs as Tasks

- Estimate work for bugs similar to tasks
- Update bug status on sprint Taskboards
- Link bugs to requirements as child items
- Drag bugs to the Planning pane to assign bugs to a sprint

> ⓘ **Note**
>
> - Bugs are assigned to the Task Category
> - User Stories (Agile), Product Backlog Items (Scrum), or Requirements (CMMI) are the natural parent work item type for Bugs
> - Bugs aren't visible on Delivery Plans

Bugs don't appear on backlogs or boards

- Manage bugs using queries

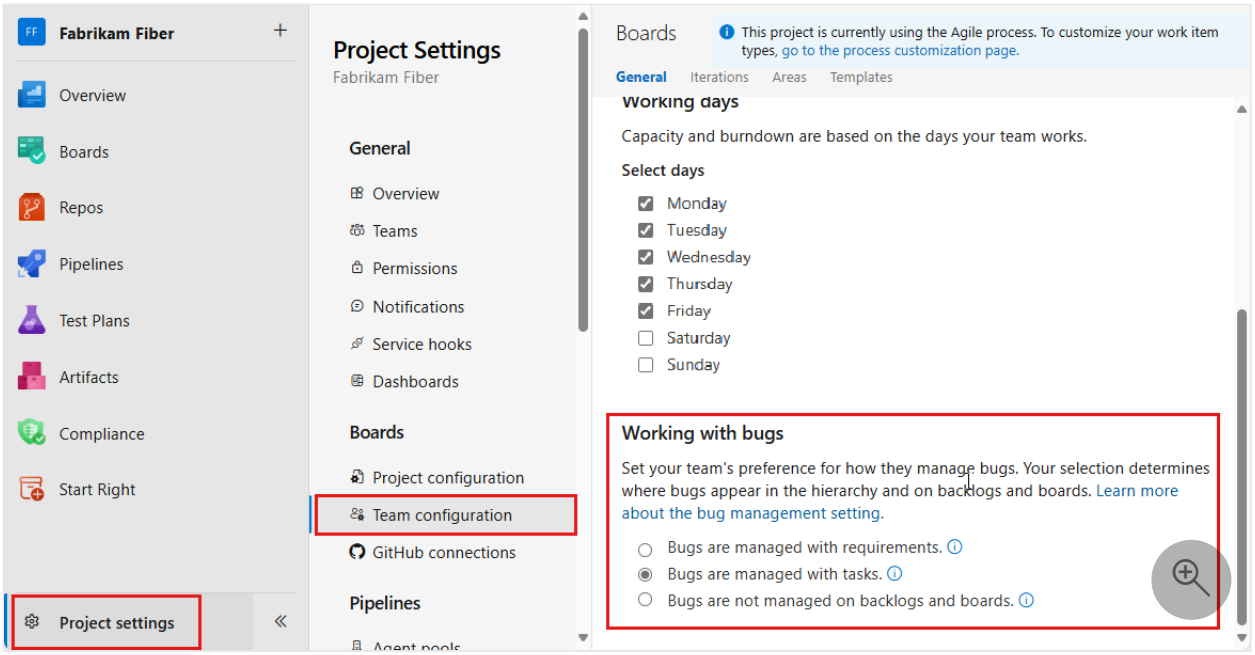
> ⓘ **Note**
>
> - Bugs are associated with the Bugs Category and don't appear on either backlogs or boards

- Bugs aren't visible on Backlogs, Boards, Sprint Backlogs, Taskboards, or Delivery Plans
- You can't drag bugs to the Planning pane to assign bugs to a sprint

---

Set your team preference for bug tracking

You can change settings from a backlog or board view, or from **Project settings** > **Team configuration**.

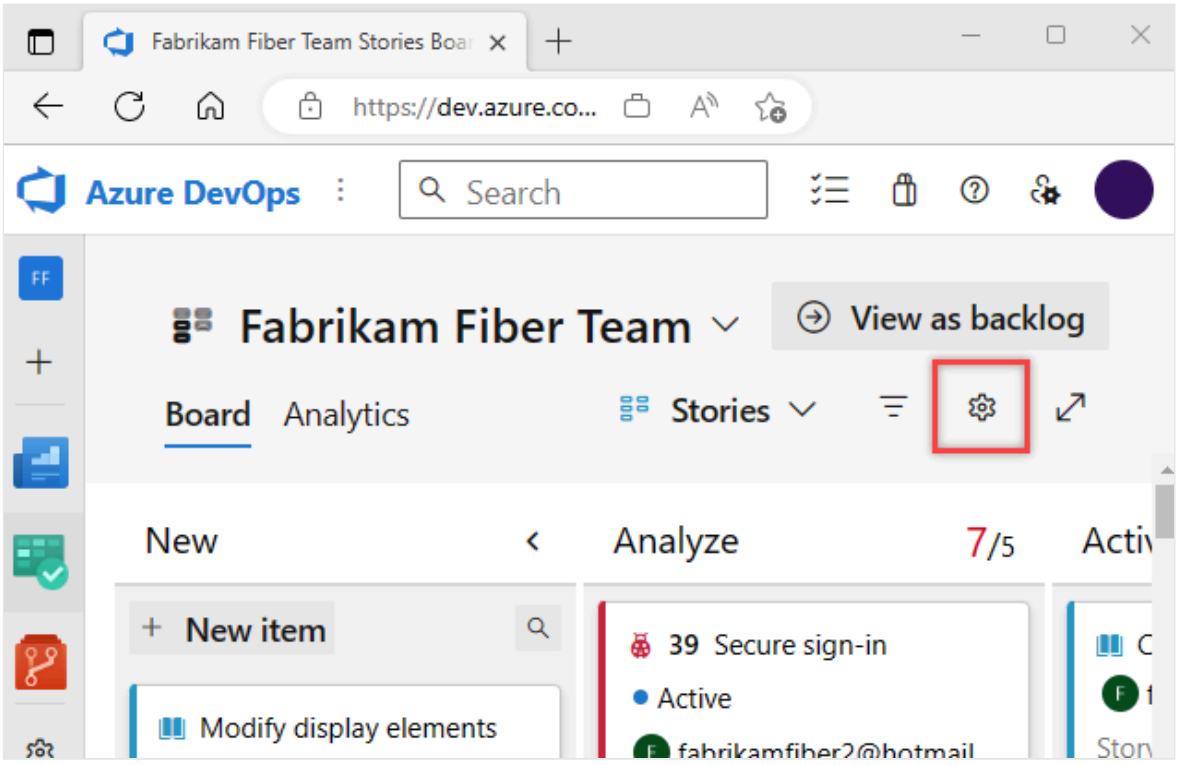


The following steps show how to change it from the board view.

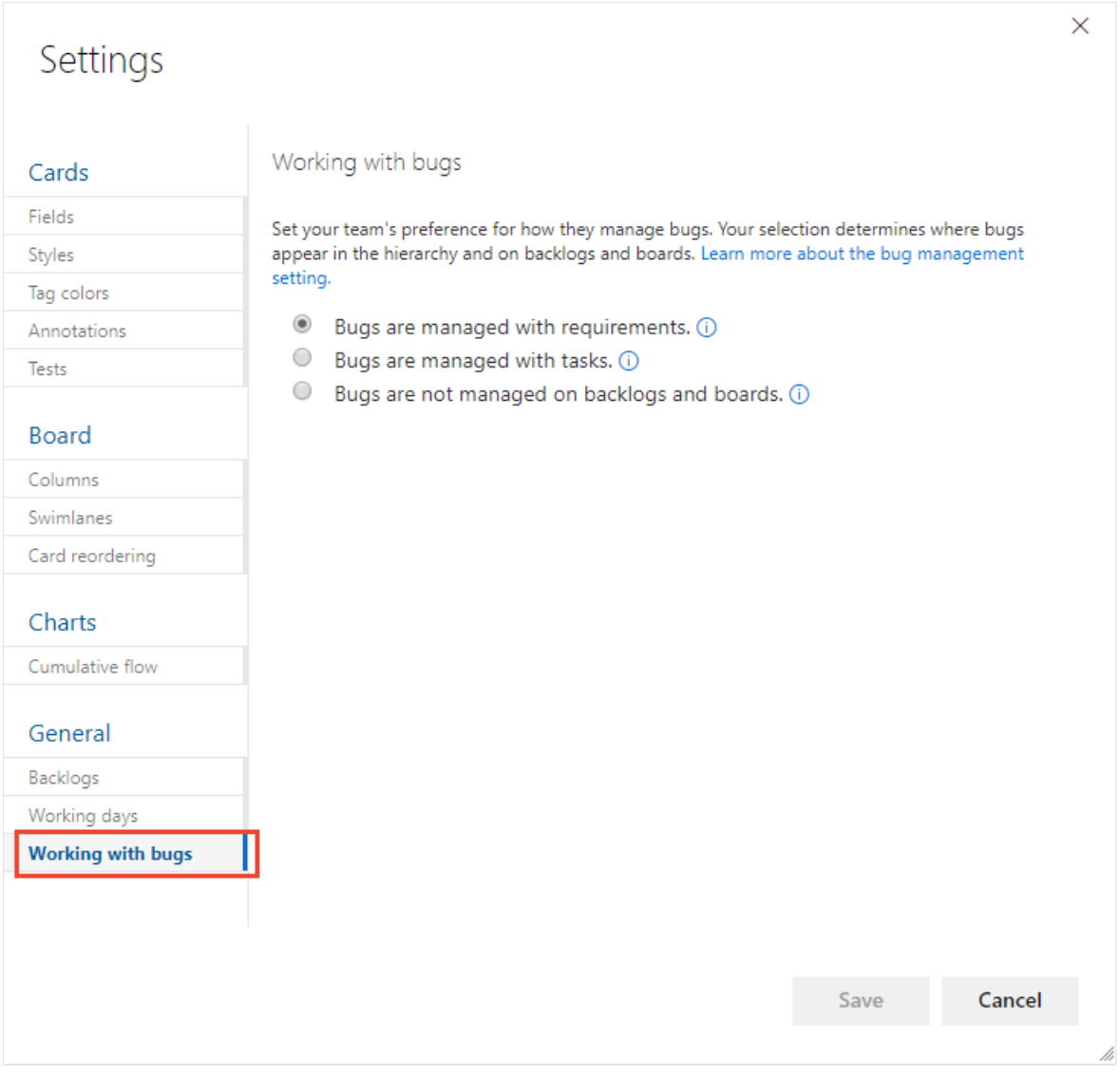
1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`) and select your project.

2. Open your board. If you're not a team administrator, get added as one. Only team and project administrators can customize the board.

3. Choose **Board settings** to configure the board and set general team settings.



4. Choose **Working with bugs** and then choose the option that best meets your team's way of working.



5. When you're done with your changes, choose **Save**.

6. To see the changes, open or refresh the team's backlog or board.

Nest items

When you manage bugs with requirements or tasks, they appear on one or more of your Agile tool backlogs and boards. However, if you nest items, by creating parent-child links of items that belong in either the Requirements or Task categories, not all items can appear on your backlogs and boards. To learn more about how nested items are treated, see Troubleshoot reordering and nesting issues.

> **Tip**
>
> If, after refreshing a backlog or board, you don't see bugs where you expect to see them, review **Troubleshoot reordering and nesting issues**. Only leaf nodes of nested items appear on the or task boards.

Add other work item types to your backlogs or boards

Bugs are a common item that teams want to track, and choose how they track them. For more information, see Manage bugs.

However, what if you want to track other work item types on your backlogs and boards?

You can add other work item types by customizing your process or project, based on the process model that you use. These items include change requests, issues, and impediments.

- For the Inheritance process model, see Customize your backlogs or boards.
- For Hosted XML and On-premises XML process models, see Add a work item type to a backlog and board.

For an overview of process models, see Customize your work tracking experience.

Create, list, and manage bugs

Bugs that are managed with requirements can be added through the product backlog or board. When bugs are managed along with tasks, you can add them to a sprint

backlog or task board. Or, capture them using other tools. For more information, see Define, triage, and manage bugs.

> **Tip**
>
> Effort should automatically be part of a bug. If you don't see it, customize the bug work item type for it to appear.

You can review bugs defined for your project by creating a query and specifying the **Work Item Type=Bug**. Or, open a predefined query: **Active Bugs** (Agile and CMMI) or **Work in Progress** (Scrum).

Related articles

- Define, capture, triage, and manage bugs
- Select backlog navigation levels for your team
- Manage teams and configure team tools
- View, run, or email a work item query
- Query by assignment or workflow changes

Feedback

Was this page helpful? Yes No

Provide product feedback

About process customization and inherited processes

Article • 07/24/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

To customize the work tracking system, you *customize* an inherited process through the administrative user interface for the organization. All projects that use an inherited process get the customizations made to that process. On the other hand, you *configure* your Agile tools—Backlogs, Sprints, boards, and Taskboards—for each team.

> **Important**
>
> To customize an on-premises project or update XML definition files to support customization, see **On-premises XML process model**. This article applies to Azure DevOps Services and Azure DevOps Server 2019 only.

There are a number of customizations you can make. The primary ones are adding custom work item types (WITs) or modifying an existing WIT to add custom fields, modify the layout, or change the workflow.

> **Note**
>
> Review changes made to an inherited process through the audit log. For more information, see Access, export, and filter audit logs.

Below you'll find an index to those tasks you can perform to customize an inherited process. Some options of inherited elements are locked and can't be customized.

> **Note**
>
> For more information, see the following articles:
>
> - About backlogs
> - Configure and customize Azure Boards
> - Create a project using the process of your choice
> - Customize your work tracking experience (process models)
> - Manage inherited processes

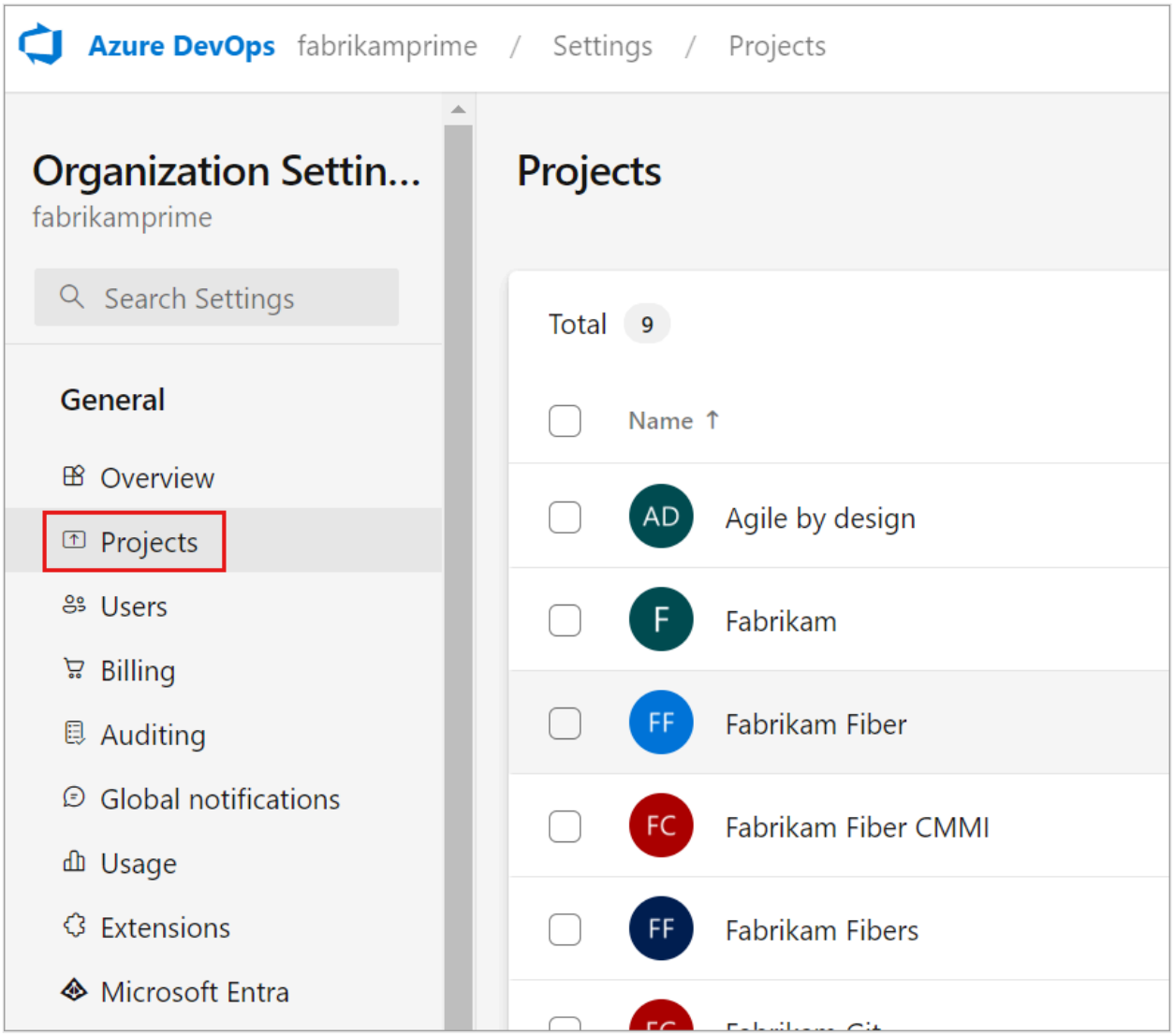
System versus inherited processes

You'll see two types of processes:

- System processes —Agile, Basic, Scrum, and CMMI—which are locked from being changed.
- Inherited processes, which you can customize and that inherit definitions from the system process from which they were created. System processes are owned and updated periodically by Microsoft. Any updates made to a system process automatically cause an update to your inherited processes and their child inherited processes. Updates to processes are documented in the Release Notes for Azure DevOps Server.

In addition, all processes are shared. That is, one or more projects can use a single process. Instead of customizing a single project, you customize a process. Changes made to the process automatically update all projects that use that process. Once you've created an inherited process, you can customize it, create projects based on it, make a copy of it, and change existing projects to use it.

For example, as shown in the following image, you see a list of projects defined for the *fabrikam* organization. The second column shows the process used by each project. To change the customizations of the *Fabrikam Fiber* project, you need to modify the *MyScrum* process (which inherits from the *Scrum* system process). Any changes you make to the *MyScrum* process also update other projects that use that process. You can't customize the *Query test* project, on the other hand, until you change it to a process which inherits from *Agile*.



Process name restrictions

Process names must be unique and 128 Unicode characters or less. Also, names can't contain the following characters: `.,;'`:~\/*|?"&%$!+=()[]{}<>`.

To rename a process, open the ... context menu for the process and choose **Edit**.

Change the reference process of a project

If you want to switch the process a project uses from one system process to another, you can do that. To make these changes, you must create an inherited process based on the process you want to switch to. For example, instructions are provided to support the following changes:

- From Basic to Agile
- From Scrum to Agile
- From Agile to Scrum

Following the guidance provided in the above listed articles, you can also make additional changes, for example, from CMMI to Agile or Agile to CMMI.

Prior to making this change, we recommend you familiarize yourself with the process you are changing to. The system processes are summarized in About processes and process templates.

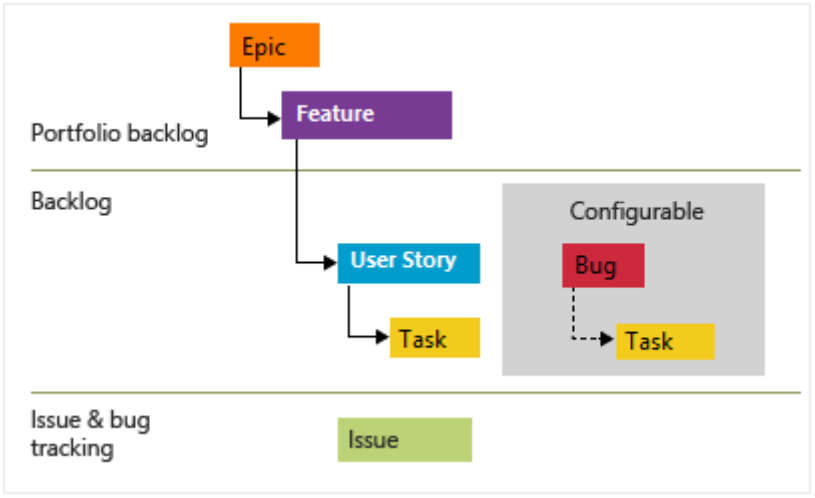
Best practices when making changes

Making changes to an inherited process is straight forward and safe. However, it is always a best practice to test those changes before applying them to an active project. Following these steps will help you surface any negative affects your process changes may have.

Inherited objects versus custom objects

Each inherited process you create inherits the WITs defined in the system process—Basic, Agile, Scrum, or CMMI. For example, the Agile process provides bug, task, user story, feature, epic, issue and test-related WITs.

Agile process



You can add fields and modify the workflow and work item form for all inherited WITs that display on the **Work Item Types** page. If you don't want users to create a WIT, you can disable it. In addition, you can add custom WITs.

Field customizations

Fields defined in the system process appear with an  inherited icon, indicating that you can make limited modifications to it in your inherited process.

Fields are defined for all projects and processes in the organization. That means that any custom field you defined for a WIT in one process can be added to any other WIT defined for another process.

Field type

Customization support

 **Inherited fields**

- Change the field label
- Show/Hide field on form
- Modify picklist (drop-down menu)
- Modify Description help text

Custom fields

- Add a custom field
- Add picklist (drop-down menu)
- Add person-name/Identity
- Add a rich-text (HTML) field
- Add a checkbox (Boolean) field
- Add a custom control
- Add custom rules to a field
- Change the field label
- Set Required/Default options
- Move the field within the layout
- Modify Description help text
- Show/Hide field on form
- Remove field from form
- Delete field

Custom control

- Add a field-level contribution or custom control

- Add a group-level or page-level contribution
- Move the control within the layout
- Show/Hide control on form

When adding custom fields, note the following limits:

- A maximum of 64 fields can be defined for each WIT
- A maximum of 512 fields can be defined per process

In addition, you can add an existing field to another WIT within the process. For example, you can add Due Date to the user story or bug WITs.

What you can't customize

- You can't change the field name or data type once you've defined it
- You can't modify the gray area on the form where the State, Reason, Area Path, and iteration path fields are located
- You can't import or define a global list as supported by the Hosted XML and On-premises XML process models. For more information, see Define global lists.

Configurable picklists

The following picklists are configured for each project and not customizable through an inherited process.

- Area paths
- Iteration paths

Picklists associated with person-name fields, such as Assigned To and Changed By, are managed based on the users you add to a project or team.

Can I rename a field or change its data type?

Renaming a field or changing the data type aren't supported actions. However, you can change the label that appears for a field on the work item form from the Layout tab. When selecting the field in a query you need to select the field name and not the field label.

Can I delete or restore a deleted field?

You can delete a field, and later restore it. Deleting a field deletes all data associated with that field, including historical values. Once deleted, you can only restore the field and recover the data using the Fields - Update REST API.

Instead of deleting a field, you may want to instead hide or remove the field from a work item form. For details, see Add and manage fields, Show, hide, or remove a field.

What is a field? How are field names used?

Each work item type is associated with 31 system fields and several more type-specific fields. You use work items to plan and track your project.

Each field supports tracking a piece of information about the work to perform. Values you assign to a field are stored in the work tracking data store which you can create queries to determine status and trends.

For descriptions and usage of each field defined for the core system processes—Scrum, Agile, and CMMI system processes—see Work item field index.

Field names

A work item field name uniquely identifies each work item field. Make sure your field names fall within these guidelines:

- Field names must be unique within the organization or project collection
- Field names must be 128 or fewer Unicode characters
- Field names can't contain any leading or trailing spaces, nor two or more consecutive spaces
- Field names must contain at least one alphabetic character
- Field names can't contain the following characters: `.,;'`:~\/*|?"&%$!+=()[]{}<>`.

Because all fields are defined for the organization, you can't add a custom field with the same field name that already exists in the organization or was added to a WIT in another inherited process.

> **Note**
>
> When you transition a project to an inherited process, you might encounter Agile tools or work items in an invalid state per the following examples:
>
> - If you designate a field as required, work items lacking that field display an error message. To proceed with further changes and save the work item,

resolve these errors.

- If you add, remove, or hide workflow states for a WIT that appears on the board, ensure you update the board column configurations for all teams defined in the project. Also, consider maintaining single ownership of work items by team area path or formalizing columns with custom states share across teams.

Custom rules and system rules

Each WIT—bug, task, user story, etc.—has several system rules already defined. Some are simple, like making the Title field required or setting a default for the Value Area field. In addition, a number of system rules define actions to take when a workflow state changes.

For example, several rules exist to copy the current user identity under the following conditions:

- When a work item is modified, copy the user identity to the Changed By field
- When the workflow state changes to Closed or Done, copy the user identity to the Closed By field.

> **ⓘ Important**
>
> Predefined system rules take precedent over any custom rule that you define which would overwrite it.

Custom rules provide support for a number of business use cases, allowing you to go beyond setting a default value for a field or make it required. Rules allow you to clear the value of a field, copy a value into a field, and apply values based on dependencies between different fields' values.

With a custom rule, you can define a number of actions based on specific conditions. For example, you can apply a rule to support these types of scenarios:

- When a value is defined for Priority, then make Risk a required field
- When a change is made to the value of Release, then clear the value of "Milestone"
- When a change was made to the value of Remaining Work, then make Completed Work a required field
- When the value of Approved is True, then make Approved By a required field

- When a user story is created, make the following fields required: Priority, Risk, and Effort

> **Tip**
>
> You can't define a formula using a rule. However, you may find a solution that fits your needs with **Power Automate** or **TFS Aggregator (Web Service) Marketplace extension**. See also **Rollup of work and other fields**.

For details on defining custom rules, see Rules and rule evaluation.

Restrict modification of select fields for select user groups

Using one of the following two conditions, you can make select fields required for a user of a security group or who are not a member of a security group.

- `current user is a member of a group...`
- `current user is not a member of a group...`

For example, you can make the Title or the State field Read-only for select users or groups.

Restrict modification of work items based on Area Path

You can disallow users from modifying select work items by setting permissions on an Area path. This is not a rule setting, but a permission setting. For more information, see Create child nodes, modify work items under an area path.

Work item type (WIT) customizations

Here are your customization options for inherited and custom WITs.

Work item type

Customization support

Inherited work item types

- Add custom rules to a WIT
- Add/remove custom fields
- Add/remove custom groups
- Add/remove custom pages
- Add/remove a custom control
- Enable/disable

Custom work item types

- Add custom WIT
- Change color or description
- Add/remove custom fields
- Add/remove custom groups
- Add/remove custom pages
- Add/remove a custom control
- Add custom rules to a wit
- Add, edit, or remove a workflow state
- Enable/disable
- Delete

What you can't customize

- You can't add or remove an inherited WIT to or from a backlog
- You can't change the position of an inherited field within the form layout (however, you can hide the field in one area of the form and add it elsewhere in the form)
- You can't remove the inherited portfolio level from the product (but you can rename them)
- You can't change the name of a custom WIT.

Work item form customizations

You can make the following customizations to a WIT form.

Group or page type

Customization support

Inherited groups

- Relabel
- Add/remove custom fields
- Show/hide fields

Custom groups

- Add, modify, re-sequence, delete
- Add/remove custom fields
- Add/Hide a group extension

Inherited pages

- Relabel
- Add/remove custom fields
- Add/remove a custom group

Custom pages

- Add, modify, re-sequence, delete
- Add/delete custom fields
- Add/hide a page extension

Layout and resizing

The web form layout is organized into three columns as shown in the image below.



If you only add groups and fields to the first two columns, then the layout reflects a two-column layout. Likewise, if you only add groups and fields to the first column, then the layout reflects a one-column layout.

The web form resizes depending on the width available and the number of columns in the layout. At maximum width, in most web browsers, each column within a page displays within its own column. As the display width decreases, each column resizes proportionally as follows:

- For three columns: 50%, 25%, and 25%
- For two columns: 66% and 33%
- For one column: 100%.

When the display width won't accommodate all columns, columns appear stacked within the column to the left.

Workflow customizations

You can customize the workflow of any work item type (WIT) by hiding inherited states or adding custom states. Inherited states vary based on the system process—Agile, Basic, Scrum, or CMMI—you selected to create your custom process.

Each default workflow for each WIT defines between two and four states and specifies the following workflow operations:

- Forward and backward transitions between each state. For example, the Basic process Issue WIT includes three states—**To Do**, **Doing**, and **Done**.
- Default reasons for each state transition

State types

Supported customizations

 Inherited states

- Hide or unhide a state
- Add rules when changing a workflow state

Custom states

- Add a workflow state
- Edit a workflow state (change color or category)
- Remove a workflow state
- Add rules when changing a workflow state

Workflow states must conform to the following rules

- Define at least one state for either the *Proposed* or *In Progress* State categories.

  > **Note**
  >
  > Before you add a workflow state, see Workflow states and state categories to learn how workflow states map to state categories.

- Define at least two workflow States.
- Define a maximum of 32 workflow States per work item type.

Unsupported workflow customizations

- Hide inherited states if you don't want them visible (you can't change their name, color, or category).
- Ensure only one state exists in the *Completed* state category. Adding a custom state to this category will remove or hide any other state.
- Keep the name of custom states as is; you can't change them.
- Use default reasons for state transitions, such as *Moved to state Triaged* and *Moved out of state Triaged*; you can't specify custom reasons.
- Accept the default location of the State and Reason fields on the form; you can't change their placement.
- Use the default state category names; you can't customize them.

Backlog and board customizations

Backlogs and boards are essential Agile tools for creating and managing work for a team. The standard backlogs—product, iteration, and portfolio—inherited from the system process are fully customizable. In addition, you can add custom portfolio backlogs for a total of five portfolio backlogs.

Backlog types

Customization support

Inherited backlogs

- Add a custom work item type (WIT)

- Add an inherited work item type
- Change the default work item type
- Rename a backlog

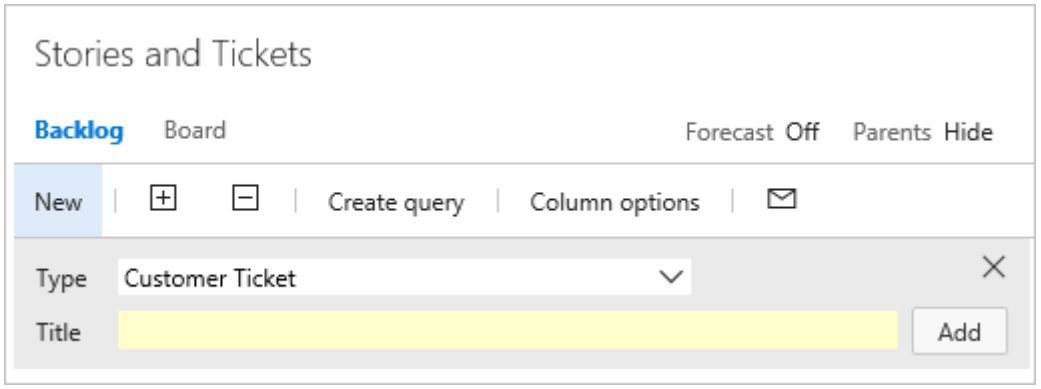
Custom portfolio backlogs

- Add a portfolio backlog that displays custom work item types (WITs)
- Edit or rename a portfolio backlog
- Delete the top-level custom portfolio backlog

Unsupported customizations:

- **Removing an inherited portfolio level:**
  - While you can't directly remove an inherited portfolio level from a product, you have a couple of options:
    - **Rename the portfolio level:** You can rename the inherited portfolio level to better suit your needs.
    - **Disable an inherited WIT:** If the inherited portfolio level includes WITs that you don't want to use, you can disable them. This action prevents teams from creating new work items of those types.
- **Inserting a backlog level:**
  - You can't insert a new backlog level within the existing set of defined backlogs. The predefined backlog levels are typically fixed (for example, Epics, Features, User Stories, Tasks), and you can't add custom ones in between.
- **Reordering backlog levels:**
  - Unfortunately, you can't reorder the backlog levels. They usually follow a predefined hierarchy, and changing their order isn't supported.
- **Adding a WIT to multiple backlog levels:**
  - Each WIT can only belong to one backlog level. You can't add a WIT to two different backlog levels simultaneously.
- **Creating a custom task backlog level:**
  - Although you can't create a custom task-specific backlog level, you can still add custom WITs to the iteration backlog. For example, you could create a custom WIT called "Enhancement" or "Maintenance" and associate it with the iteration backlog.
- **Managing bugs:**
  - The Bug WIT doesn't belong to any specific backlog level by default. Instead, each team can decide how they want to manage bugs. You can choose to show bugs on backlogs and boards or handle them separately.

When you change the default WIT for a backlog level, it causes that WIT to appear by default in the quick add panel. For example, *Customer Ticket* appears by default in the following quick add panel for the product backlog.



Object limits

For a list of limits placed on the number of fields, WITs, backlog levels, and other objects you can customize, see Work tracking object limits.

Feedback

Was this page helpful? Yes No

Provide product feedback

List work item fields and attributes

Article • 11/22/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Work item fields are used to track information within Azure DevOps. These fields are defined at the organization level and are shared across all projects in the organization. To review the fields defined for your organization, you can use one of the following two tools, available for both Inherited and Hosted XML process models:

- Process > Fields web page
- Work Item Field Explorer

For a description of each field defined with a system process, see Work item field index.

Prerequisites

Permissions: To view the fields defined for an organization or collection, be a member of the **Project Collection Valid Users** application group or have the **View instance-level information** permission set to **Allow** for the organization or collection.

List or review fields

To list or review fields, use one of the following tools based on your process model—**Inherited**, **Hosted XML**, or **On-premises XML**. For an index of fields defined within the default processes, see Work item field index.

Expand table

Tool	Inheritance	Hosted XML	On-premises XML
Web portal: List inherited and custom-defined fields	✓	✓1	
Work item field explorer	✓	✓	✓
witadmin listfields command line tool	✓	✓	✓

> **Note**

1. Only supported for default processes (Agile, CMMI, Scrum).

Field data types and names

Each work item type specifies the fields included in work items of that type. Each field has multiple attributes, many of which are system-defined and immutable.

Fields get defined by the following three attributes:

- **Data type**: Indicates the kind of data that can be entered into the field, such as Boolean, Double, Integer, HTML, or String. For detailed descriptions of each data type, see Query fields, operators, and macros.
- **Friendly name**: The user-friendly name assigned to the field, used when selecting a **Field** in a query clause. This name might differ from the name displayed on the work item form.
- **Reference name**: The identifier used when creating WIQL queries, improvised work item templates, executing REST API commands, or defining XML work item type definitions. The reference name is permanent and can't be changed once set.

For a detailed description of each field attribute and instructions on how to list them, see Field attributes and List field attributes later in this article. For an overview of work item types (WITs) and work items, refer to Track work with user stories, issues, bugs, features, and epics.

What is a field? How are field names used?

Each work item type includes 31 system fields and several type-specific fields. Work items are used to plan and track your project.

Each field captures specific information about the work to be performed. The values you assign to these fields are stored in the work tracking data store, enabling you to create queries that determine status and trends.

For descriptions and usage of each field defined for the core system processes—Agile, Basic, Scrum, and CMMI—refer to the Work item field index.

Field names

A work item field name uniquely identifies each work item field. Ensure your field names adhere to the following guidelines:

- **Uniqueness**: Field names must be unique within the account or project collection.
- **Length**: Field names must contain 128 Unicode characters or fewer.
- **Spacing**: Field names can't have leading or trailing spaces or contain two or more consecutive spaces.
- **Composition**: Field names must include at least one alphabetic character.
- **Prohibited Characters**: Field names can't include the following characters: `.,;'`:~/*|?"&%$!+=()[]{}<>`.

Since custom fields are defined at the organization or collection level, you can't add a custom field with the same name to multiple processes.

For more information, see Naming restrictions and conventions.

System and predefined fields

A work item field name uniquely identifies each field within Azure DevOps. Ensure your field names comply with the following guidelines:

- **Uniqueness**: Field names must be unique within an account or project collection.
- **Length**: Field names must be 128 Unicode characters or fewer.
- **Spacing**: Field names can't have leading or trailing spaces or contain multiple consecutive spaces.
- **Composition**: Field names must include at least one alphabetic character.
- **Prohibited Characters**: Field names can't include the following characters: `.,;'`:~/*|?"&%$!+=()[]{}<>`.

Since custom fields are defined at the organization or collection level, you can't add a custom field with the same name to multiple processes.

For more information, see Naming restrictions and conventions.

Custom fields

Because custom fields are defined at the organization or project collection level, you can't add a custom field with the same name to multiple processes.

When adding custom fields, note the following limits:

- **Maximum fields per work item type (WIT):** 64
- **Maximum fields per process:** 512

The field data type determines the kind and size of data that can be stored in the field. Each field can have only one type defined within a project collection, encouraging the

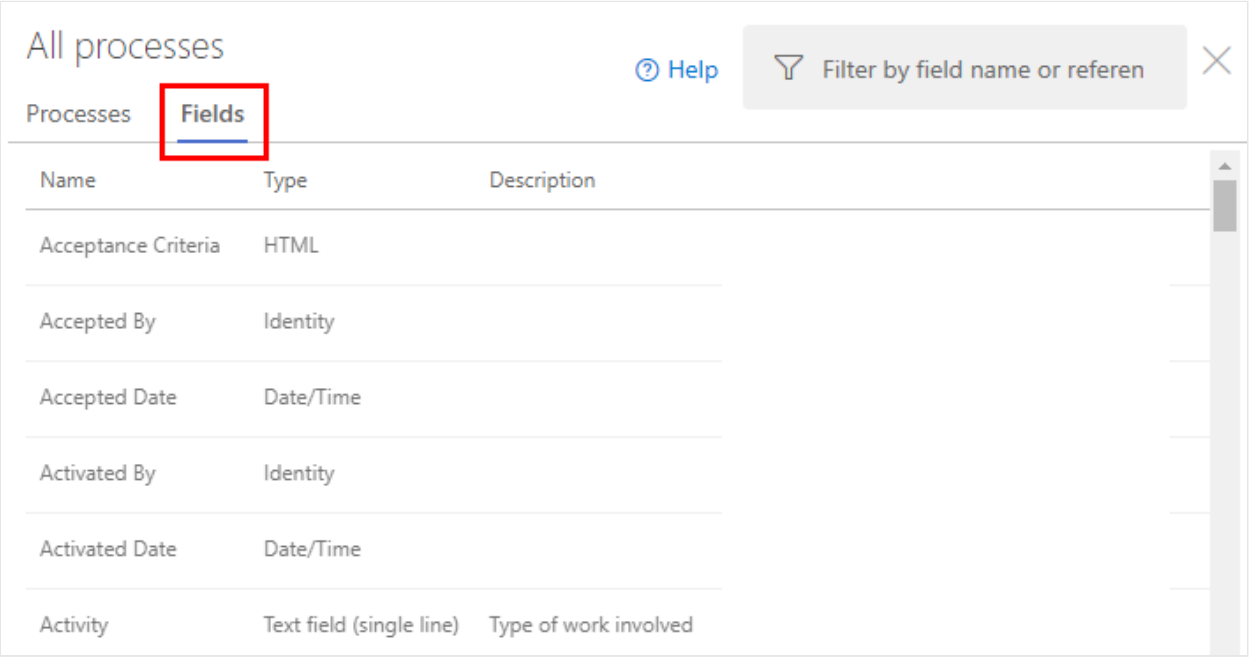
use of common fields across different projects and work item types.

When you add a custom field to an inherited process, Azure DevOps assigns a reference name prefixed with *Custom* followed by the field name without spaces. For example, adding a field named *DevOps Triage* results in the reference name **Custom.DevOpsTriage**. Spaces aren't allowed in reference names.

For more information, see Naming restrictions and conventions.

How can I determine the field data type?

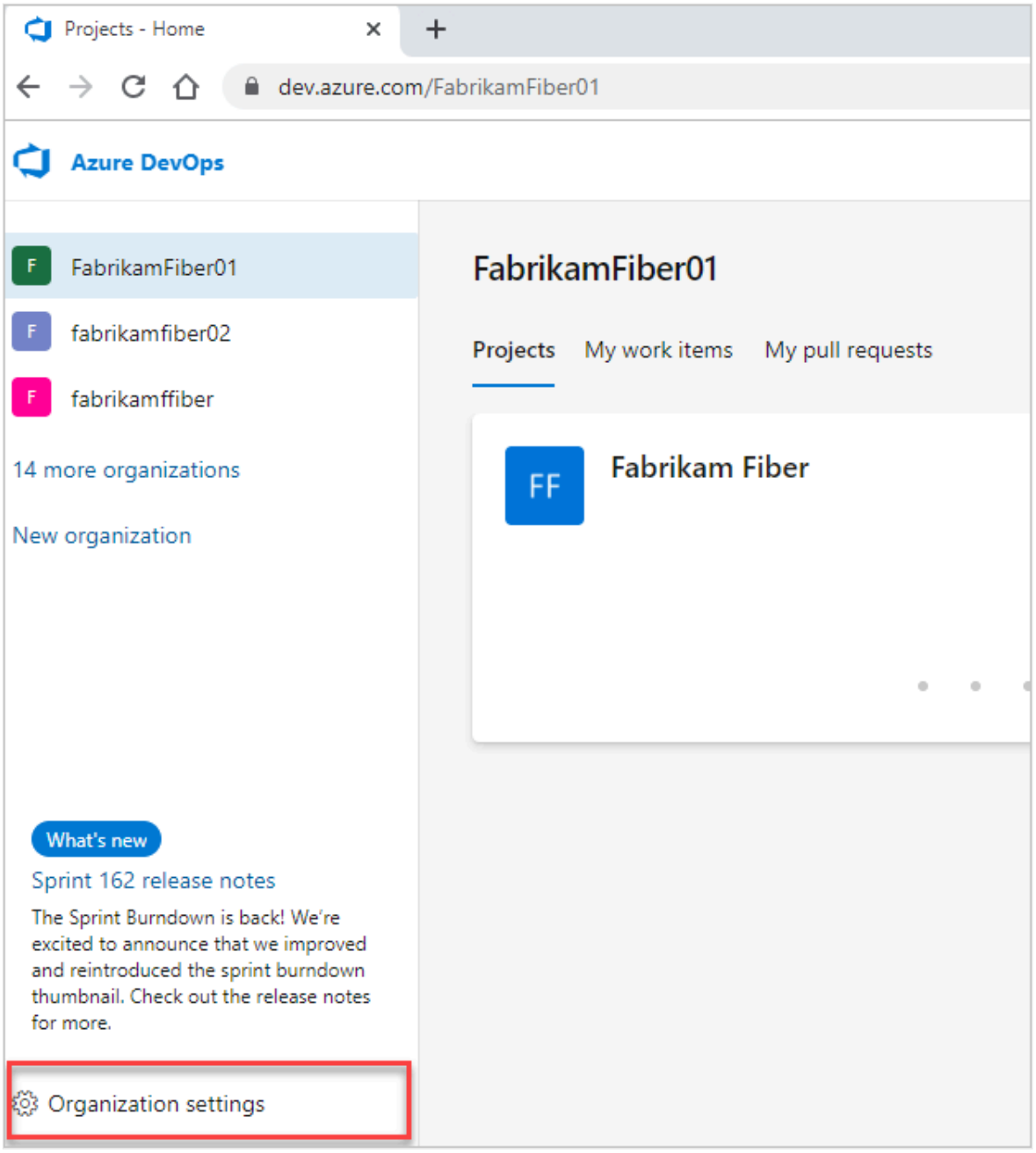
You can view the data type of fields defined for your organization by opening the Process>Fields page.



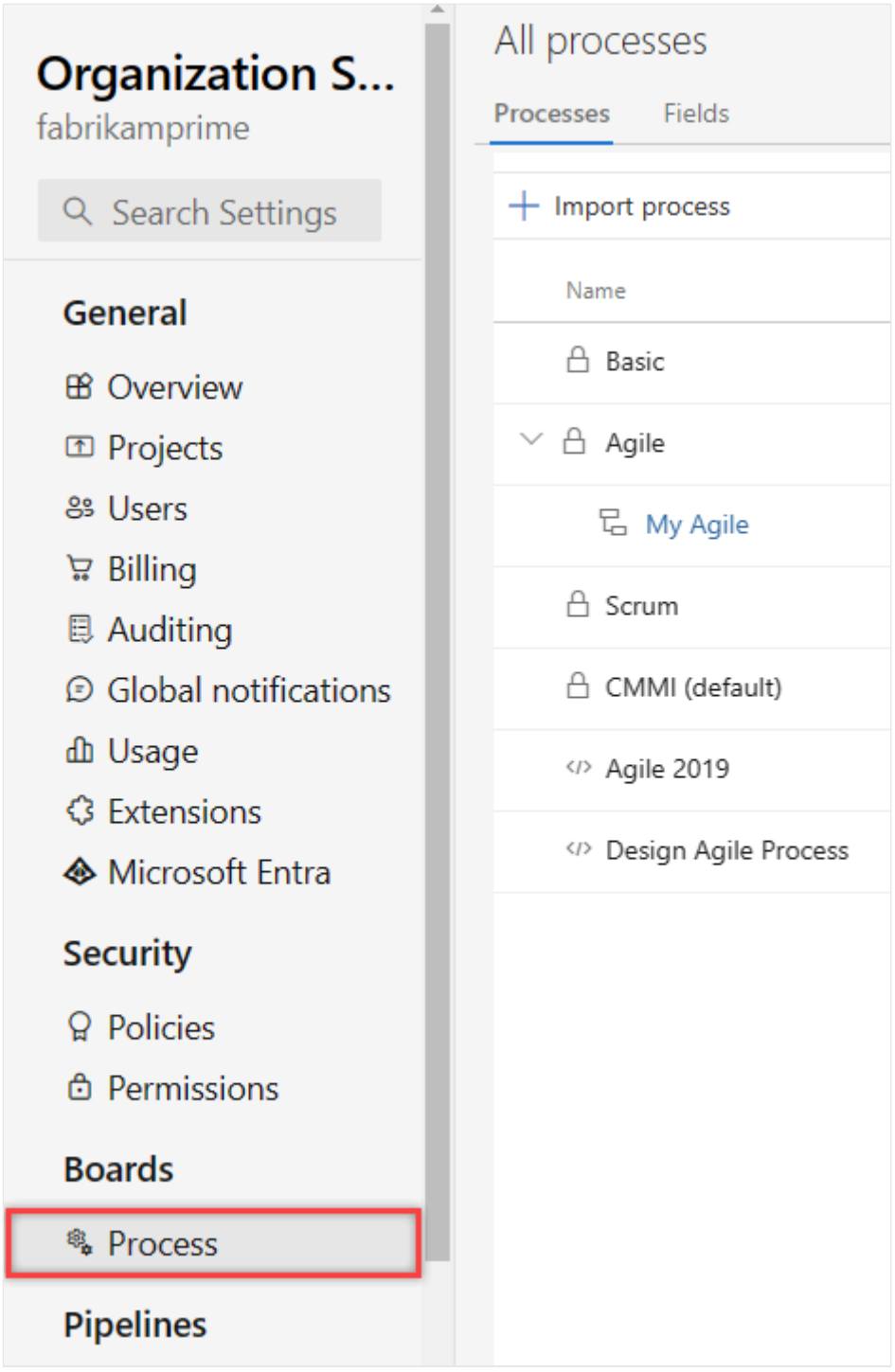
Review fields list

To review the list of fields defined for an organization or collection, do the following steps:

1. Select the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.

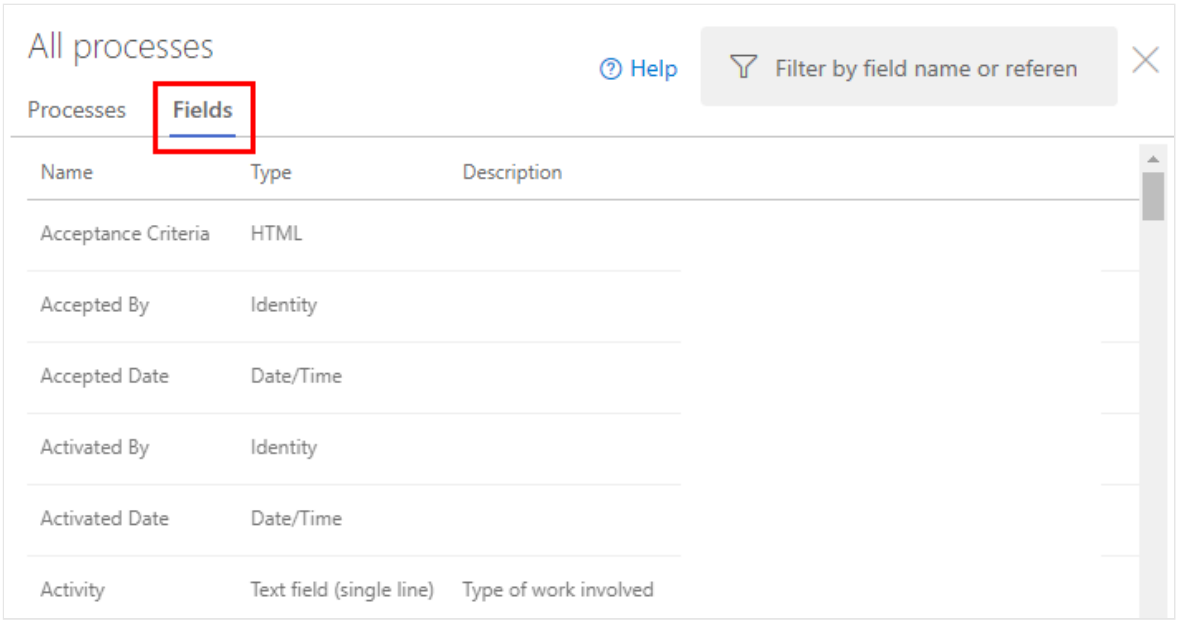


2. Select **Process**.



3. Select **Fields**.

   Fields listed correspond to all fields defined for the organization or collection, which includes all custom fields and the fields defined for system processes.

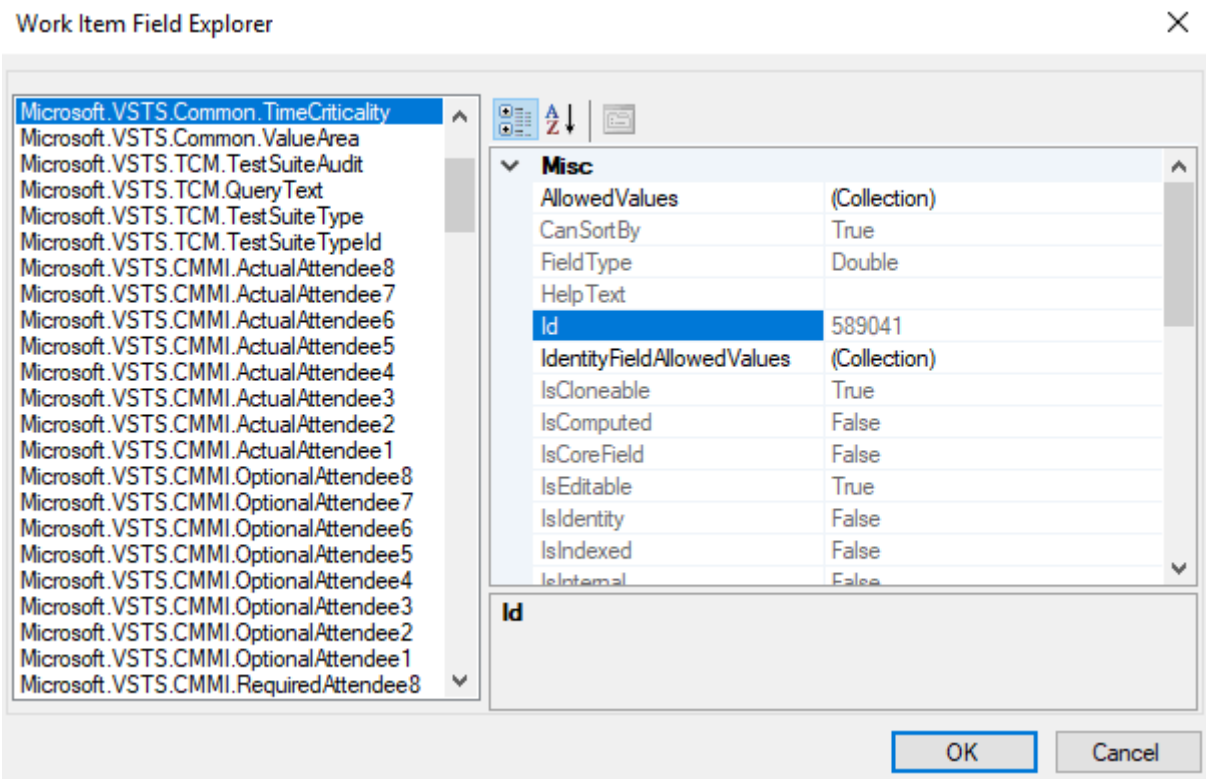


> **Note**
>
> If you don't see **Fields**, then your collection uses the On-premises XML process. The Fields page isn't supported for that process.

For descriptions, usage, and reference names of each field, refer to the Work item field index. Additionally, you can retrieve field reference names using the Work Item Types Field - List REST API.

Work Item Field Explorer

You can look up the assignments of field attributes using the Work Item Field Explorer tool.



To access the Work Item Field Explorer, install the Process Editor Tool. Based on your installed version of Visual Studio, get the Process Editor Tool from one of the following extensions.

- **Visual Studio 2019 & 2022:** As of now, there's no specific Process Template Editor extension available. But, you can still customize process templates using other tools such as any text editor or XML editor. You can also use the Azure DevOps web portal for certain customizations, especially for inherited processes.
- **Visual Studio 2017**: TFS Process Template Editor. You can also use this version of the Process Editor to modify the old-style work item forms. You can't use it to edit forms associated with the new web forms.
- **Visual Studio 2015**: TFS Power Tools.

Field attributes

There are many nonchangeable and hidden attributes for each work item field. The following table describes each attribute. Attributes have different names depending on whether you retrieve them through the **Fields - Get REST API** or view them through the Work Item Field Explorer (**WIFE**) tool.

Attributes assigned to a field depend on the platform and version you're using. For example, some attributes aren't supported with the Inheritance process. To look up the reference name for a field, see the Work item field index.

Attribute

Attribute type

Description

REST:
WIFE: **AllowedValues**

collection

Gets the collection of valid values for a field that contains picklist values. You can change this by specifying a picklist or global list (on-premises).
Can change?=Yes

REST: **canSortBy**
WIFE: **CanSortBy**

boolean

Indicates whether you can sort query results with this field.
Can change?=No

REST: **description**
WIFE: **HelpText**

string

Specifies a description for the field, which also defines the help text that appears when you hover over the field within the work item form.
Can change?=Yes

REST:
WIFE: **ID**

Integer

Specifies the internal ID of the field.
Can change?=No

REST:
WIFE: **IsCloneable**

boolean

Indicates whether the value defined for the field is copied when a user chooses to copy a work item. For example, **Title**, **Tags**, and **Description** fields are copied, but the **ID** and **History** fields aren't copied.
Can change?=No

REST:
WIFE: **IsComputed**

boolean

Indicates if the value set by this field is computed by the system (True) or not (False). Examples of computed fields are ones that are set by the system, such as the **ID**, **Revised Date**, **Changed Date**, and **External Link Count**.
Can change?=No

REST:
WIFE: **IsCoreField**

boolean

Indicates whether this field is specified for all work item types.
Can change?=No

REST:
WIFE: **IsEditable**

boolean

Indicates if users can modify this field (True) or not (False). Examples of noneditable fields are ones that are set by the system, such as the **ID**, **Revision**, **Created By**, and **Changed By** fields
Can change?=No

REST: **isIdentity**
WIFE: **IsIdentity**

boolean

Indicates whether this field is an *Identity* field. Identity fields are string fields used to store user identities.
Can change?=No

REST:
WIFE: **IsIndexed**[1]

boolean

Indicates whether this field is indexed to support search.
Can change?=No

REST:
WIFE: **IsLongText**

boolean

Indicates that the field can contain more than 255 characters, such as fields assigned a data type of *PlainText*, *HTML*, or *History*.
Can change?=No

REST: **isPicklist**[2] WIFE:

Boolean

Indicates whether the field is associated with a picklist. The value is set to *True* when a custom field is defined for Azure DevOps and Picklist (String) or Picklist (Integer) type is selected. The value is set to *False* for inherited fields that define picklists.
Can change?=No

REST: **isPicklistSuggested**[2] WIFE:

Boolean

Indicates whether the field allows users to enter their own values for a picklist. The value is set to *True* when a custom field is defined for Azure DevOps, Picklist (String), or Picklist (Integer) type is selected, and the checkbox for **Allow users to set their own values** is checked.
Can change?=Yes

REST: **isQueryable**
WIFE: **IsQueryable**

boolean

Indicates if the field shows up within the set of fields you can add to filter a work item query (True), or not (False). Most fields are queryable.

Can change?=No

REST:
WIFE: **IsReportable** [3]

boolean

Indicates if the reportable attribute is defined or set to anything other than **None**. This attribute can be changed for on-premises environments.
Can change?=Yes

REST:
WIFE: **IsUsedInGlobalWorkflow**

boolean

Indicates if the field is defined within a global workflow.
Can change?=No

REST:
WIFE: **IsUserNameField**

boolean

Indicates if the field is used to display an Identity field.
Can change?=No

REST: **name**
WIFE: **Name**

string

Friendly name assigned to the field. The friendly name can't be changed for Azure DevOps, but can be changed for on-premises using the **witadmin changefield** command.
Can change?=On-premises only

REST: **picklistId**
WIFE: **HelpText**

GUID

If the field is a picklist, the identifier of the associated picklist, otherwise null. A unique GUID value is assigned when a custom field is defined for Azure DevOps and Picklist

(String) or Picklist (Integer) type is selected.
Can change?=No

REST:
WIFE: **ProhibitedValues**

collection

Gets the collection of prohibited values for a field that specifies such values. You can only define prohibited values for on-premises deployments.
Can change?=On-premises only

REST: **readOnly**
WIFE:

Boolean

Indicates whether the field is set to read only. For Azure DevOps Services, only custom fields can be changed to be read-only. System fields can't be modified.
Can change?=Yes

REST: **referenceName**
WIFE: **ReferenceName**

string

Specifies the reference name of a field.
Can change?=No

REST:
WIFE: **ReportingAttributes**[3]

Specifies **Detail**, **Dimension**, or **Measure**, depending on whether and how you want the field to be included in reports. Data from fields that have a value other than **None** for this attribute are exported to the data warehouse and can be included in SQL reports.
Can change?=On-premises only

REST:
WIFE: **ReportingName**[3]

string

Specifies the label for a field when data appears in SQL reports. If you don't specify a value, the field's friendly name is used.
Can change?=On-premises only

REST:
WIFE: **ReportingReferenceName**[3]

string

Specifies a different reference name to a field that is used when data is exported to the relational data warehouse. If you don't specify a value, the fields reference name is used.
Can change?=On-premises only

REST: **supportedOperations**
WIFE:

set

The set of query operators that are valid for use when referencing this field. For a quick reference of supported operations based on data type, see Query quick reference, Operators, and macros supported for each data type.
Can change?=No

REST:
WIFE: **SupportsTextQuery**

boolean

Indicates whether the field supports text queries such as **Contains Words**, **Does Not Contains Words**.
Can change?=No

REST:
WIFE: **SystemType**

string

Specifies the data type of the field, referencing the system name such as System.DateTime or System.String.
Can change?=No

REST: **type**
WIFE: **FieldType**

string

Specifies the data type of the field, such as *Boolean*, *DateTime*, *Integer*, *String*, and so on. For a complete list and descriptions, see Query fields, operators, and macros.
Can change?=No

REST: **usage**
WIFE: **Usage**

string

Specifies whether the field is intended for use with work items (*WorkItem*) or work item link (*WorkItemLink*) objects. The usage for most fields is WorkItem. For a complete list of usage values, see Get Fields, FieldUsage.
Can change?=No

> **Note**
>
> 1. For on-premises deployments, you can enable indexing for a field to improve query response times when filtering on the field. For more information, see Indexed fields later in this article.
> 2. The **isPicklist** and **isPicklistSuggested** attributes are only assigned to custom fields defined for an inherited process. The Inherited process model is supported for Azure DevOps Server 2019 and later versions. For more information, see Inherited process model.
> 3. All reporting attributes are valid only for on-premises deployments whose projects have been configured to support SQL Server Reporting and SQL Server Analysis Services.

List field attributes

You can list the attributes assigned to a field by using the **Fields - Get REST API**. Replace *OrganizationName* with your actual organization name.

```
REST
```

```
https://dev.azure.com/OrganizationName/_apis/wit/fields/FieldReferenceName
```

For example, to list the attributes for the Iteration Path, use the reference name `System.IterationPath` for the `fabrikam` organization:

REST

```
https://dev.azure.com/fabrikam/_apis/wit/fields/System.IterationPath

**Returned data:**

> [!div class="tabbedCodeSnippets"]
> ```JSON
> {
> "name": "Iteration Path",
> "referenceName": "System.IterationPath",
> "description": "The iteration within which this bug will be fixed",
> "type": "treePath",
> "usage": "workItem",
> "readOnly": false,
> "canSortBy": true,
> "isQueryable": true,
> "supportedOperations": [
> {
> "referenceName": "SupportedOperations.Under",
> "name": "Under"
> },
> {
> "referenceName": "SupportedOperations.NotUnder",
> "name": "Not Under"
> },
> {
> "referenceName": "SupportedOperations.Equals",
> "name": "="
> },
> {
> "referenceName": "SupportedOperations.NotEquals",
> "name": "<>"
> },
> {
> "referenceName": "SupportedOperations.In",
> "name": "In"
> },
> {
> "name": "Not In"
> }
> ],
> "isIdentity": false,
> "isPicklist": false,
> "isPicklistSuggested": false,
> "url": "https://dev.azure.com/mseng/_apis/wit/fields/System.IterationPath"
> }
```

```
> ```
```

Add and modify fields

To add fields to a process, you add them to one or more work item types. For more information, see Customize an inheritance process.

Related articles

- Refer to the Query quick reference to quickly understand query syntax and usage.
- Explore the Work item field index to view all available work item fields.
- Add and manage fields for an inherited process to customize fields according to your inherited process requirements.
- Access the metadata reference for Azure Boards Analytics to understand analytics metadata and reporting capabilities.

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Feedback

Was this page helpful? Yes No

Provide product feedback

Add a custom field to a work item type (Inheritance process)

Article • 06/04/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

You add a custom field to support tracking data requirements you have that aren't met with the existing set of fields. For example, you can add a custom field to track a customer ticket number. For a list of all fields defined for your organization—which includes all fields defined for system and inherited processes—see View work item fields and attributes. For a description of each field defined with a system process, see Work item field index.

For more information, see About process customization and inherited processes.

> **Tip**
>
> To customize a single project, always start by **creating an inherited process and updating your project to that process**. Then, all the customizations that you make to the inherited process automatically appear for the project you migrated.

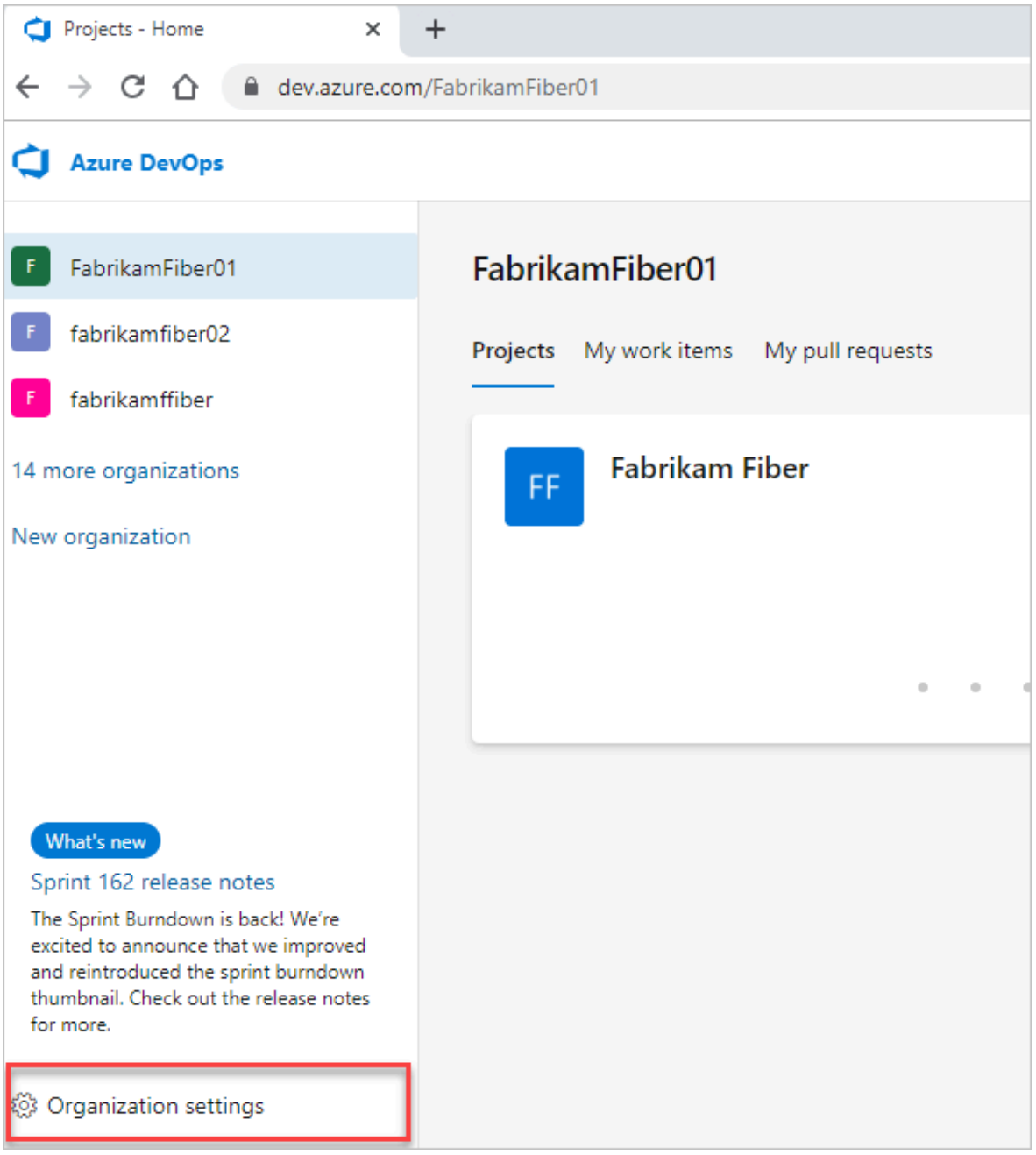
Prerequisites

We recommend checking out Configure and customize Azure Boards, which offers guidance on tailoring Azure Boards to align with your specific business requirements.

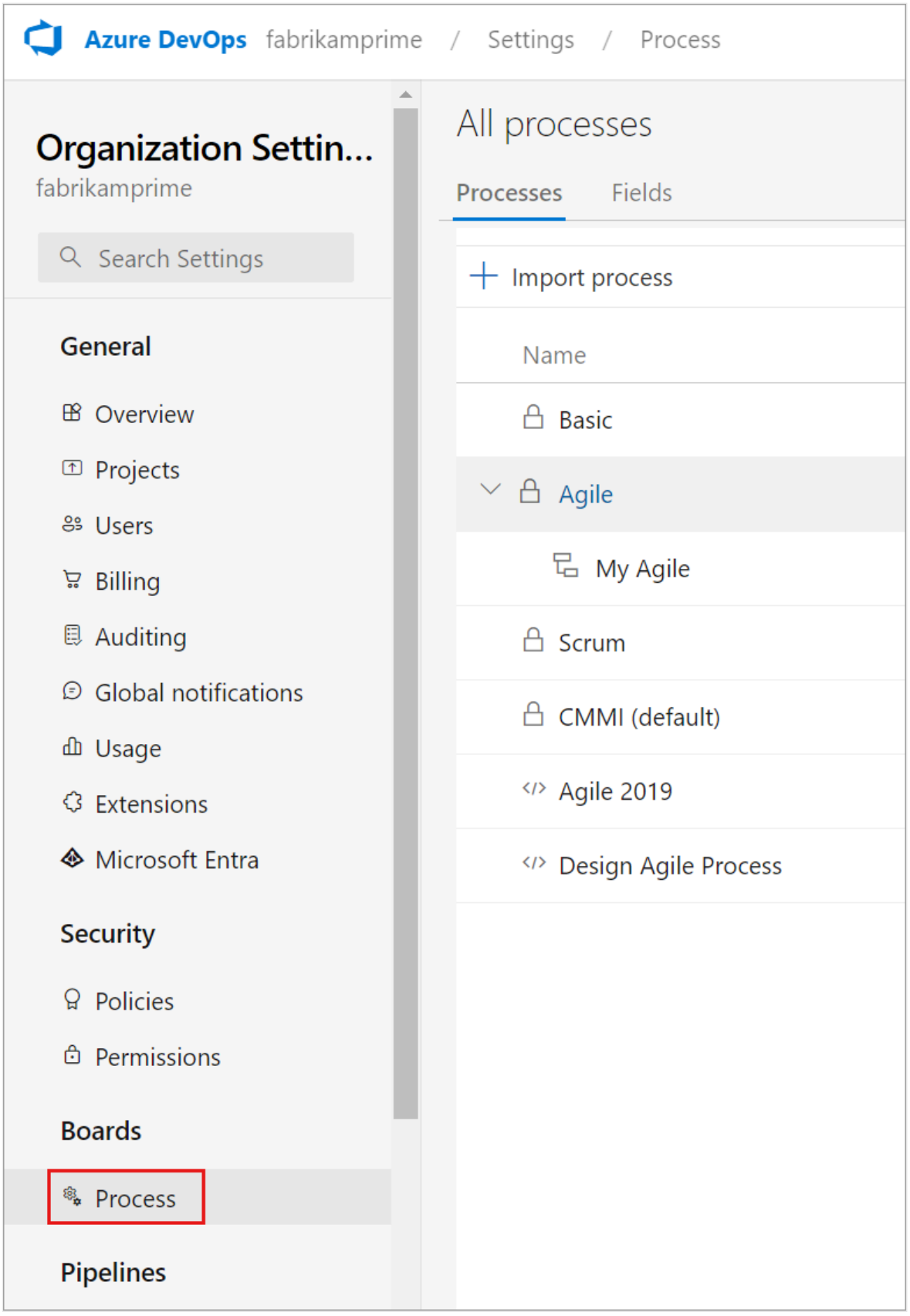
- You must have an organization in Azure DevOps.
- To create, edit, and manage processes, you must be a member of the **Project Collection Administrators** group, or have the corresponding collection-level permissions **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**. For more information, see Set permissions and access for work tracking, Customize an inherited process.

Open Settings > Process

1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`).
2. Select **Organization settings**.



3. Select **Process**.



Create inherited process

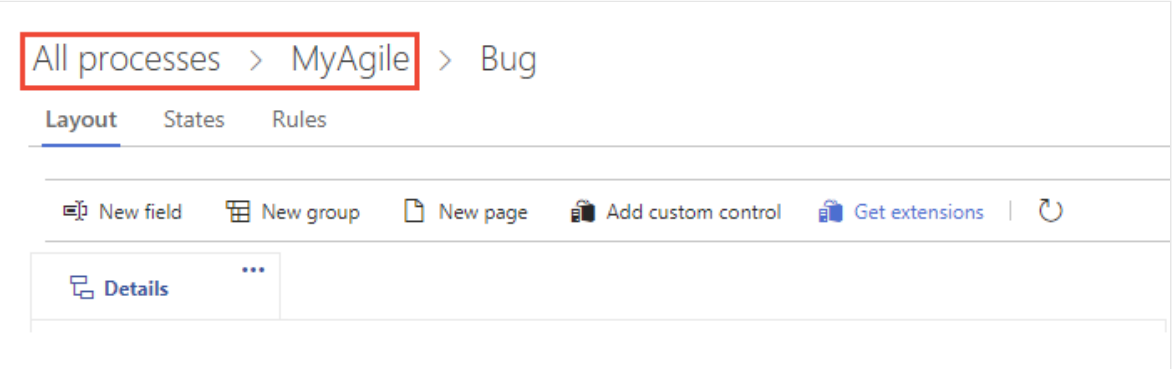
For more information, see Create inherited process.

Add a field

Do the following steps to add a field to an existing work item type. You can't use the **Steps** control on a custom work item type based on the Test Case work item type.

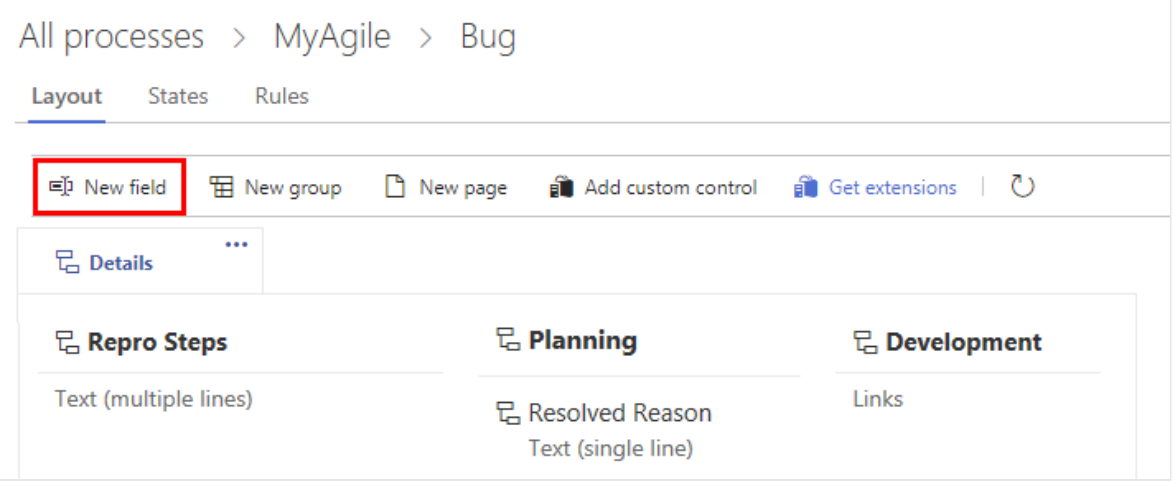
1. From the **Process** page of the selected inherited process, choose the work item type you want to add the custom field to.

   Here we choose the Bug WIT. The breadcrumb links allow you to move back to the **All Processes** and **MyAgile** process page.

   If the **New field** and other options are disabled, you don't have the necessary permissions to edit the process. See Set permissions and access for work tracking, Customize an inherited process.

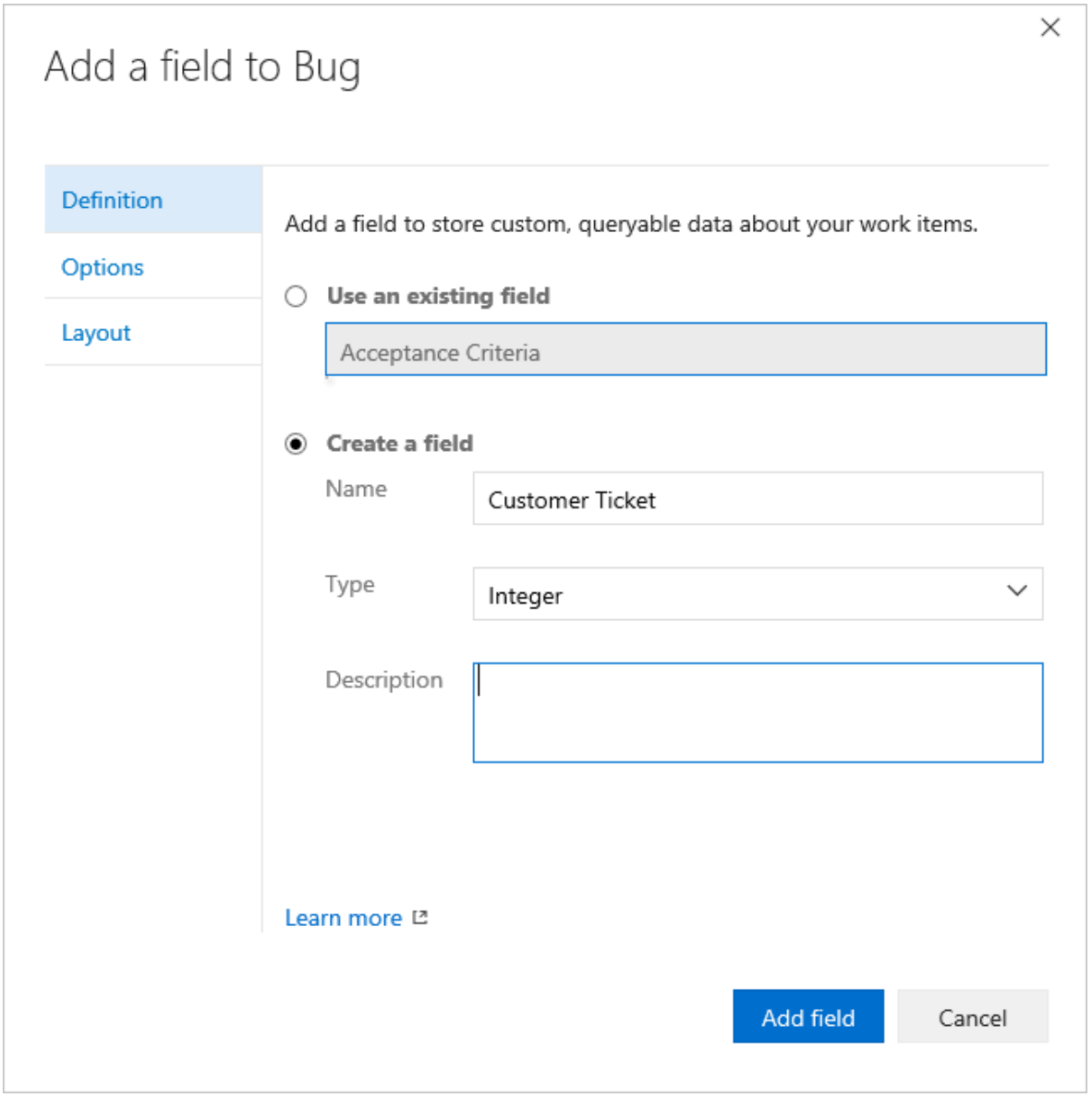
2. With the WIT selected, choose the New field **New field**.

3. Name the field and select the field type from one of the supported data types. Field names must be unique and no more than 128 characters. For additional restrictions, see What is a field? How are field names used?. Optionally, add a Description.

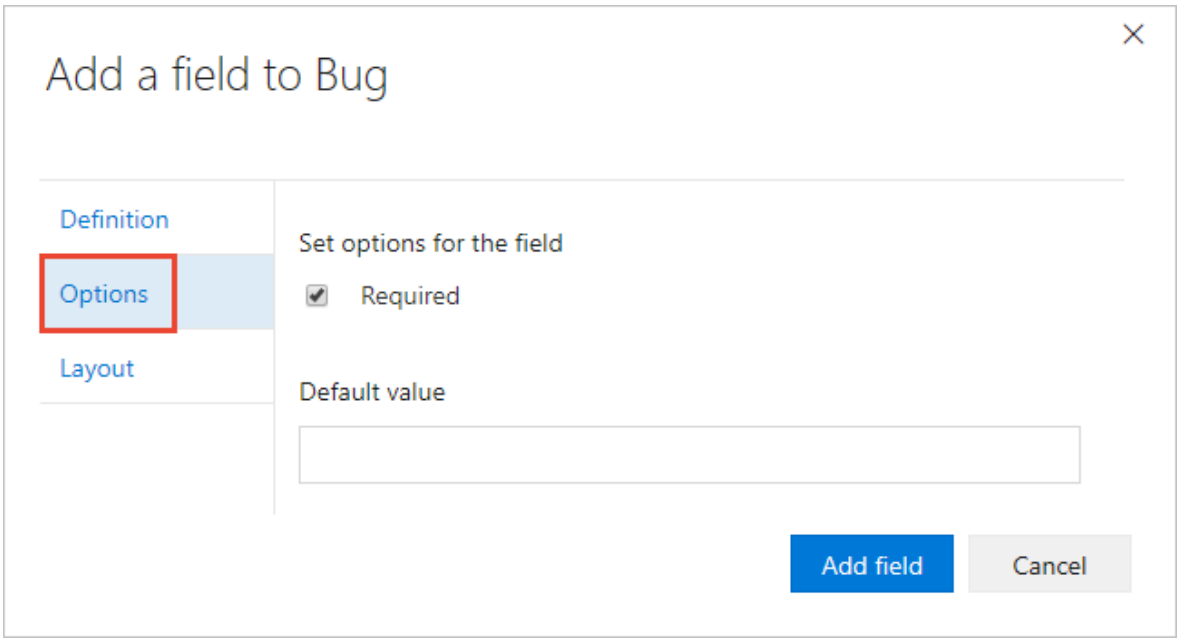
   > **Note**
   >
   > Once created, you can't change the field name or data type.

Here we add an Integer field labeled Customer Ticket.



Additional data types you can add include: Picklist, Identity, Rich-text, HTML, and Checkbox.

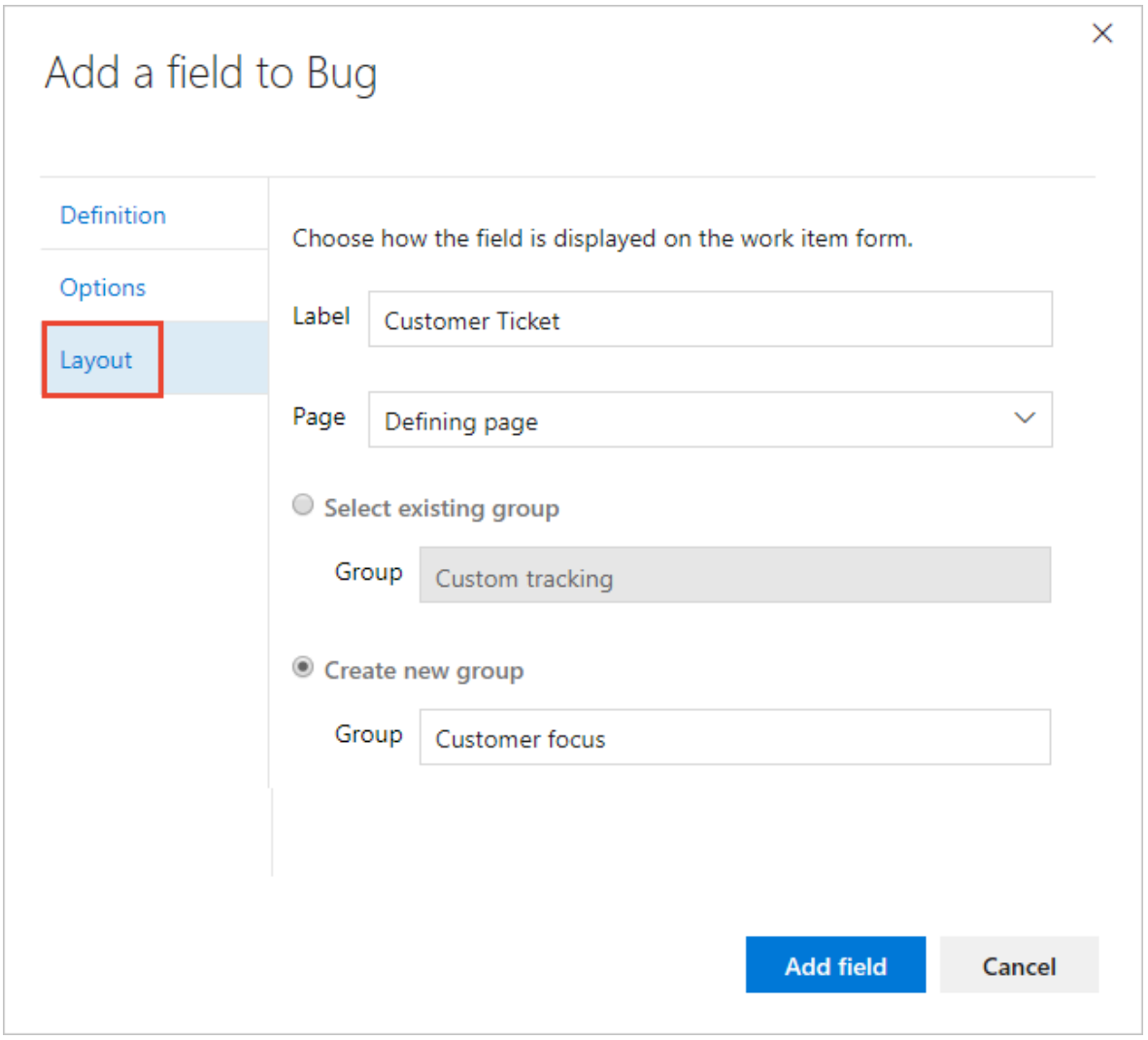
4. (Optional) On the **Options** tab, indicate if the field is required and specify a default value. Or leave these blank.



By making a field Required, users must specify a value for the field in order to save it. The default value you specify is set when you create a work item as well as every time a work item is opened and the field is empty.

5. (Optional) On the **Layout** tab, you can enter a different form label than the name of the field. Also, you can choose the page and group where the field appears on the form.

   Here, we add the Customer Ticket field to a new group labeled Customer focus.



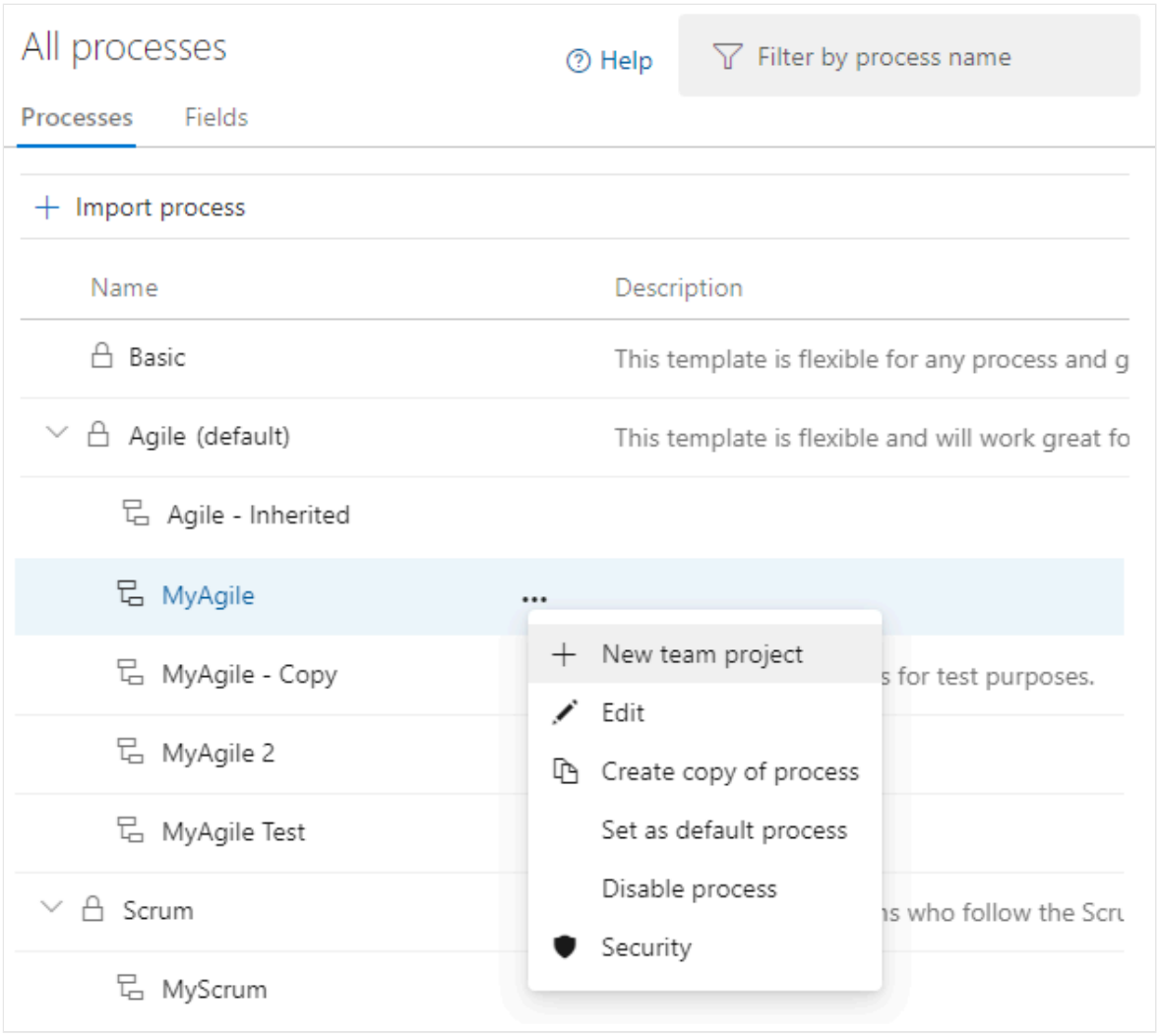
6. Choose **Add field** to complete adding the field. If you haven't specified its layout location, it is added to the first group of fields on the layout form.

> **Tip**
>
> Once you've added a field, you can drag-and-drop it within a page to relocate it on the form. If you have several fields you want to add to a custom page or group, then you may want to add those elements first and then add your fields. You can't add a field to the gray area within the form where the Assigned To, State, and Reason fields are located.

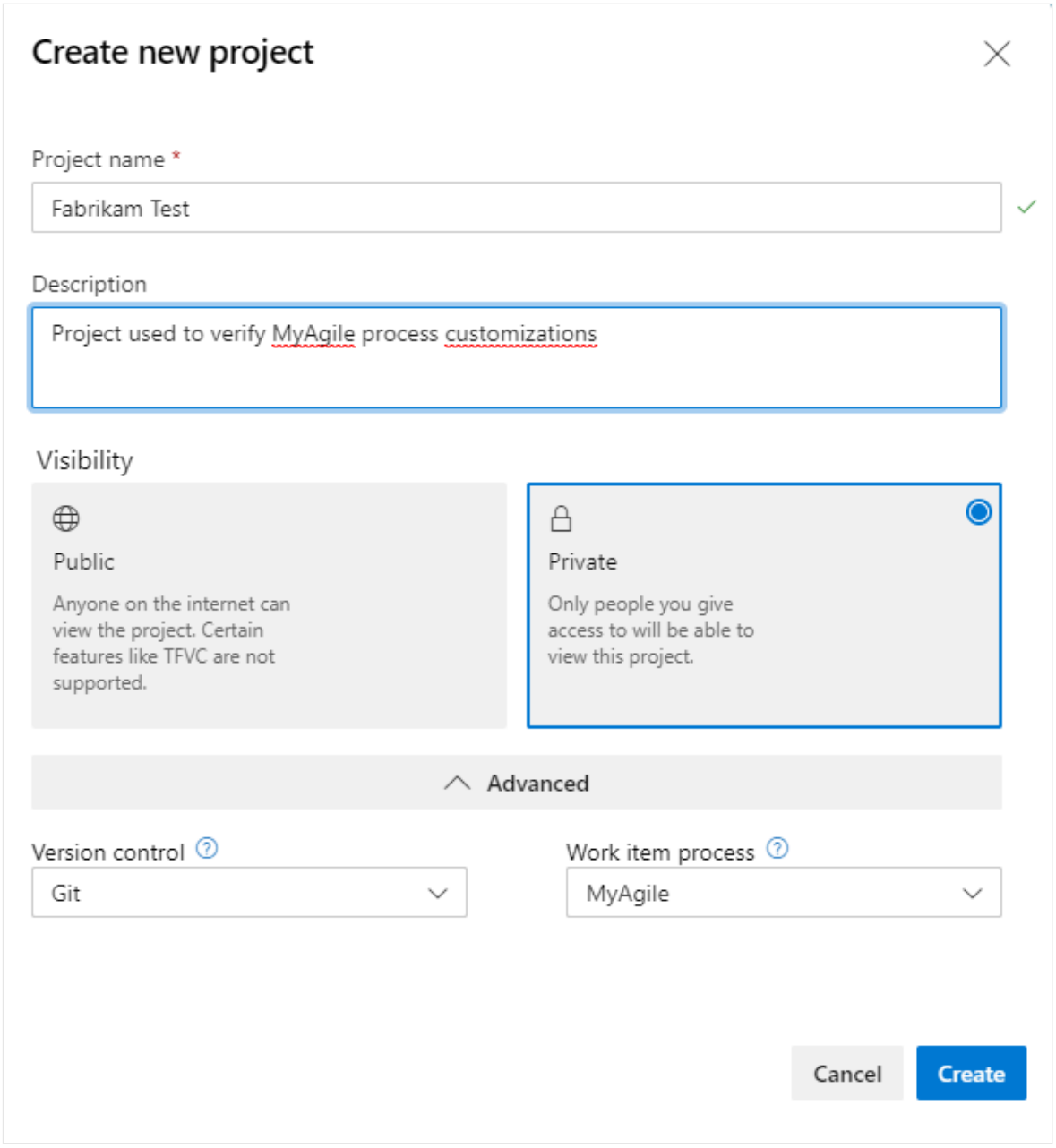
Verify the customization you made

We recommend that you create a test project and apply your customized inherited process to it to verify the changes you've made.

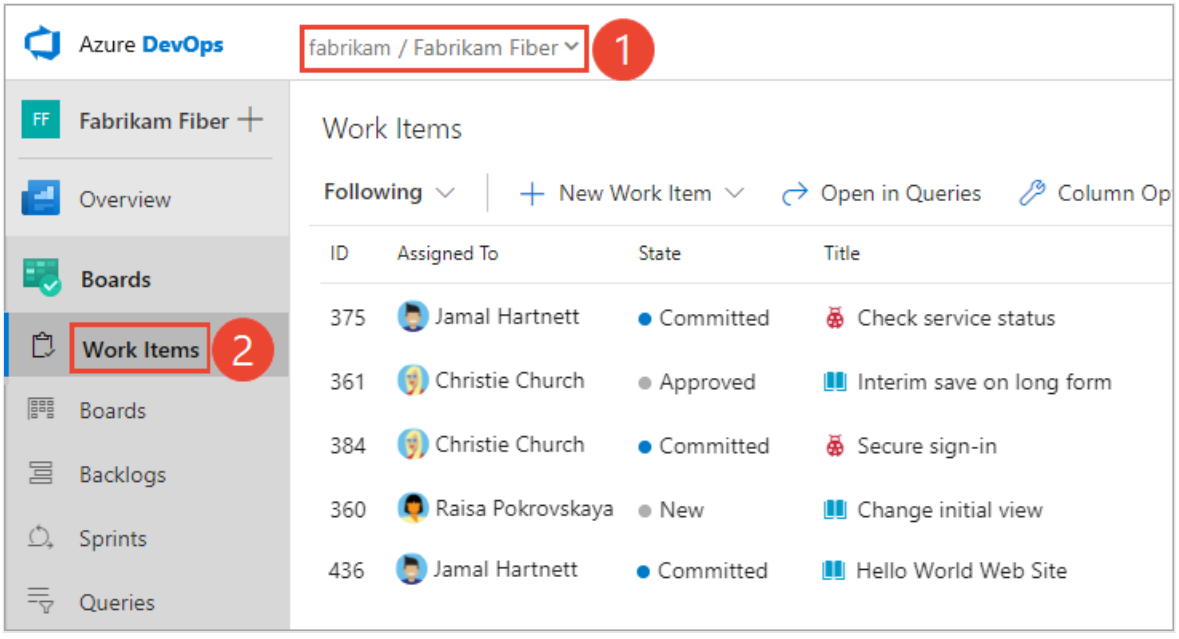
1. Open the **All processes** page, and choose the ... context menu for the process you want to use, and then select **New team project**.



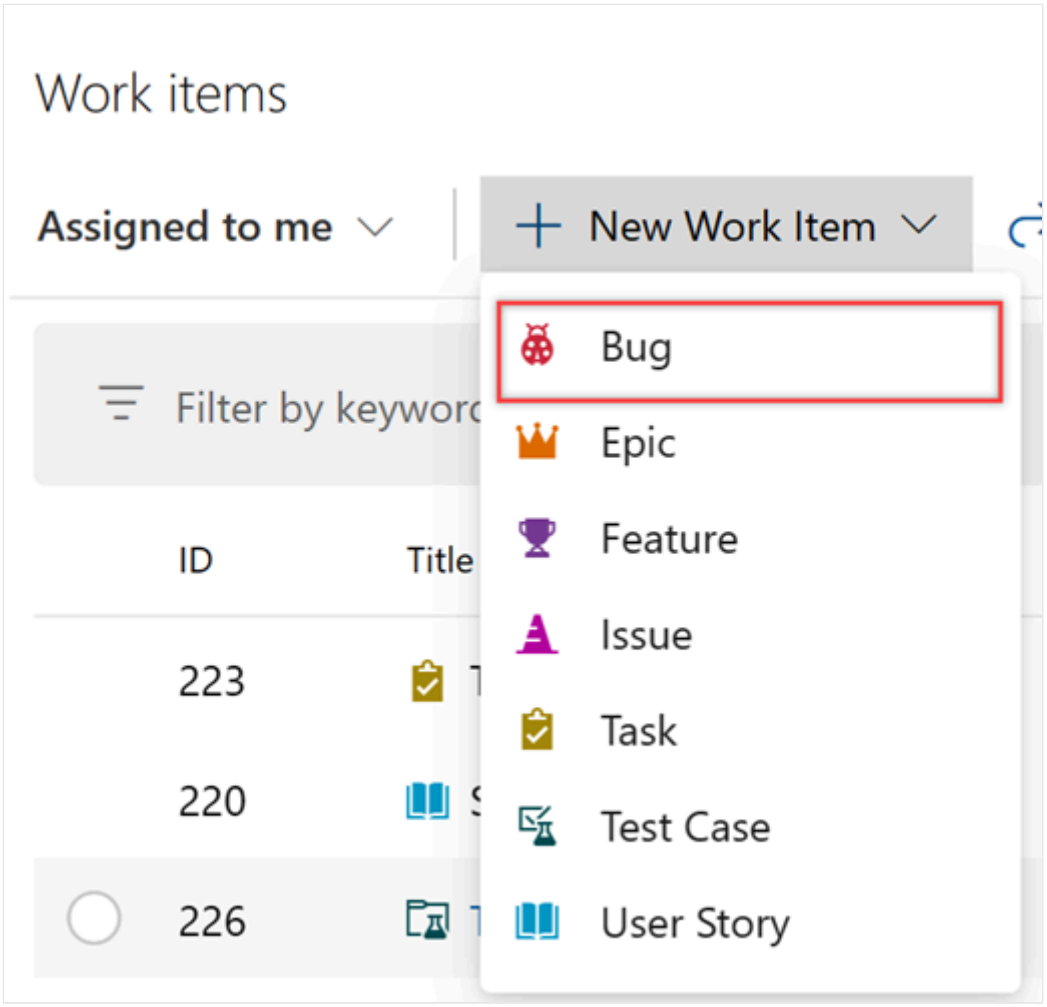
2. The Create new project page opens. Fill out the form.



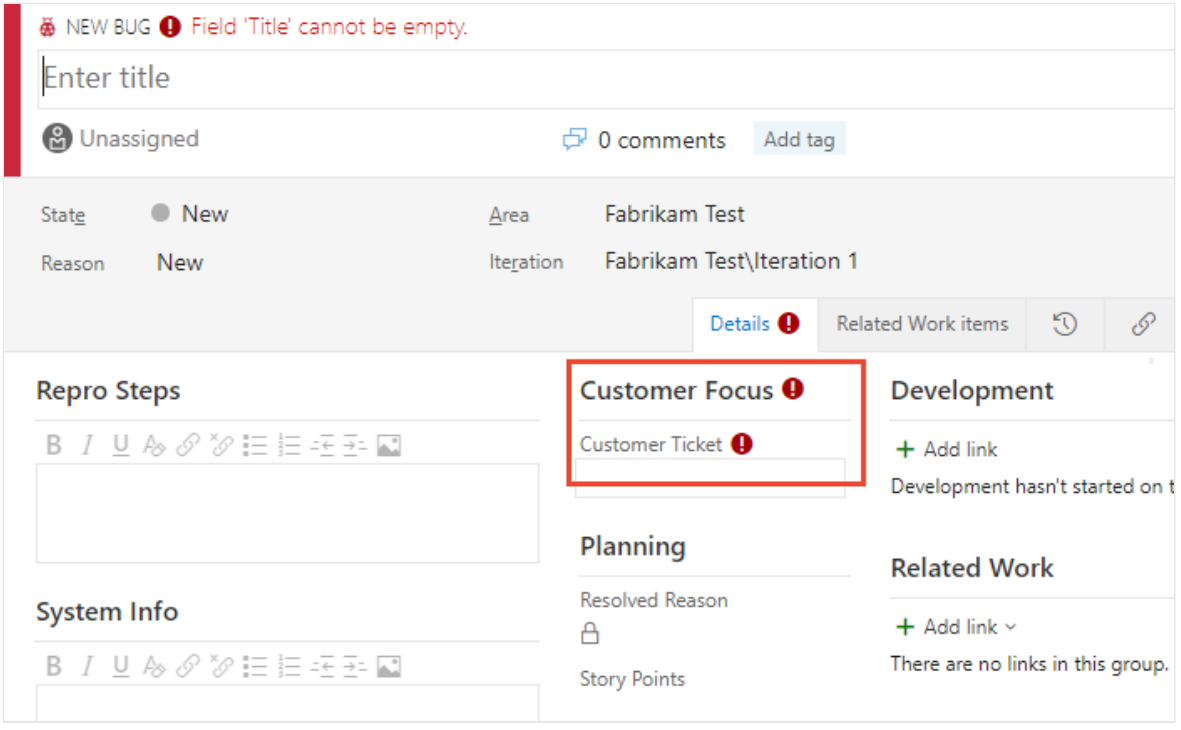
3. Open **Work Items**. (1) Check that you have selected the right project, then (2) choose **Work>Work Items**.



4. Select the WIT you customized. Here we choose **Bug**.



5. Verify that the field you added appears on the form. Note that the ❗ (exclamation mark) icon indicates the field is required.



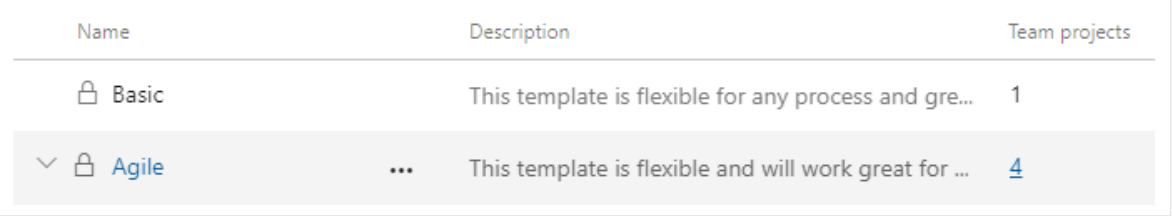
Apply the customized process to your project

After you've verified your customizations, you can now apply the process to your existing project.

> **Tip**
>
> As you customize a WIT, all projects that reference the inherited process that you're customizing automatically updates to reflect the custom WITs you've added. To view your customizations, refresh your web browser.

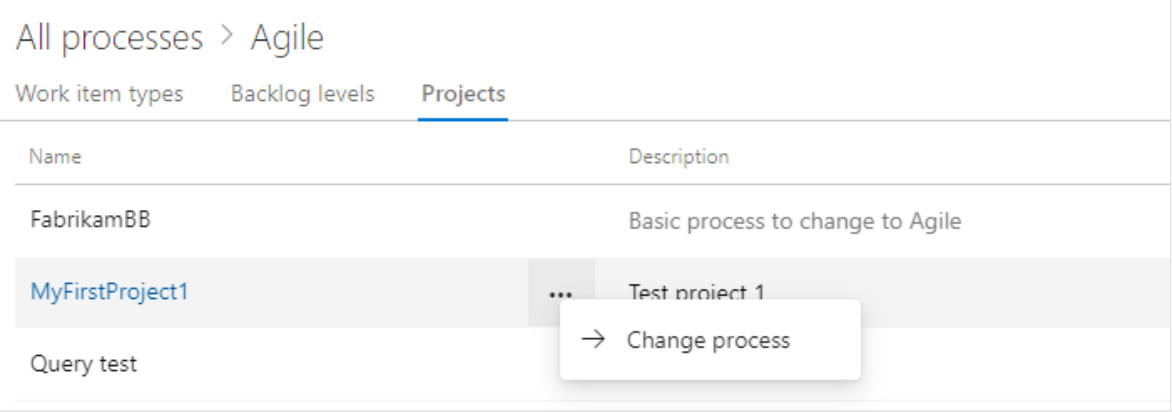
1. For the process currently used by the project, choose the number of projects.

   Here we open the menu for the Agile default process.

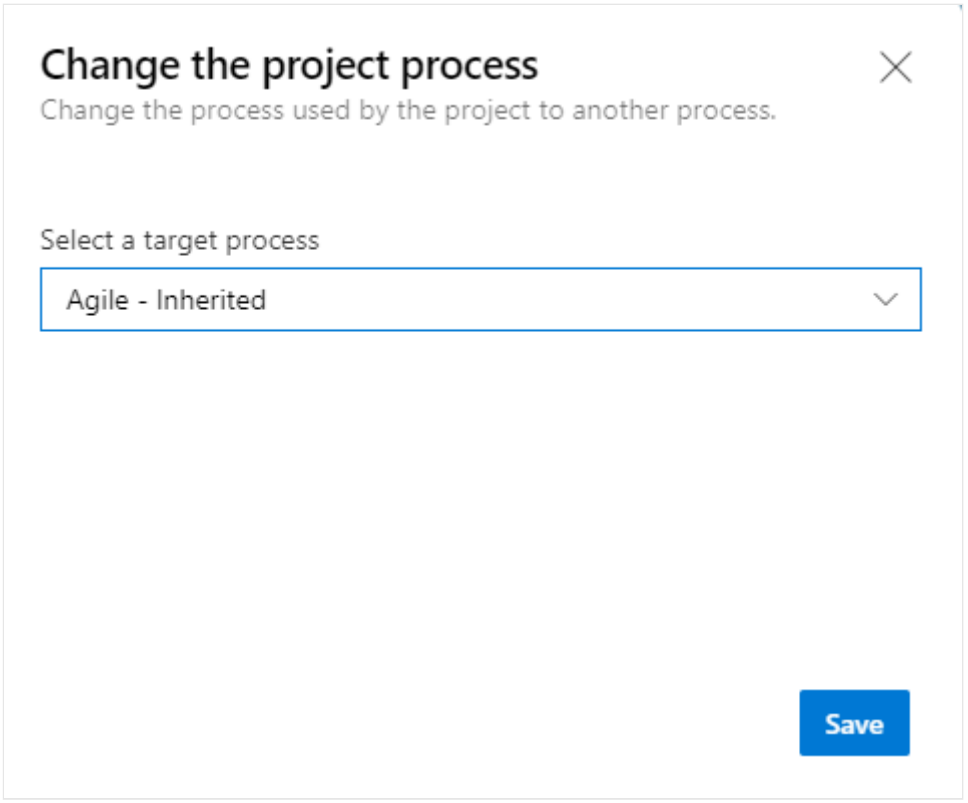
   

2. Open the ... context menu for the project you want to change, and choose the **Change process** option.

   Here we open the menu for the MyFirstProject1.

3. From the Change the project process dialog, choose the process from the menu of options. And, then choose **Save**.



Next steps

Add a custom work item type Or Add or manage fields

Related articles

> **Note**
>
> Review changes made to an inherited process through the audit log. For more information, see Access, export, and filter audit logs.

- View work item fields and attributes
- Add and manage fields for an inherited process
- Customize a project using an inherited process
- Create and manage inherited processes

Feedback

Was this page helpful? Yes No

Provide product feedback

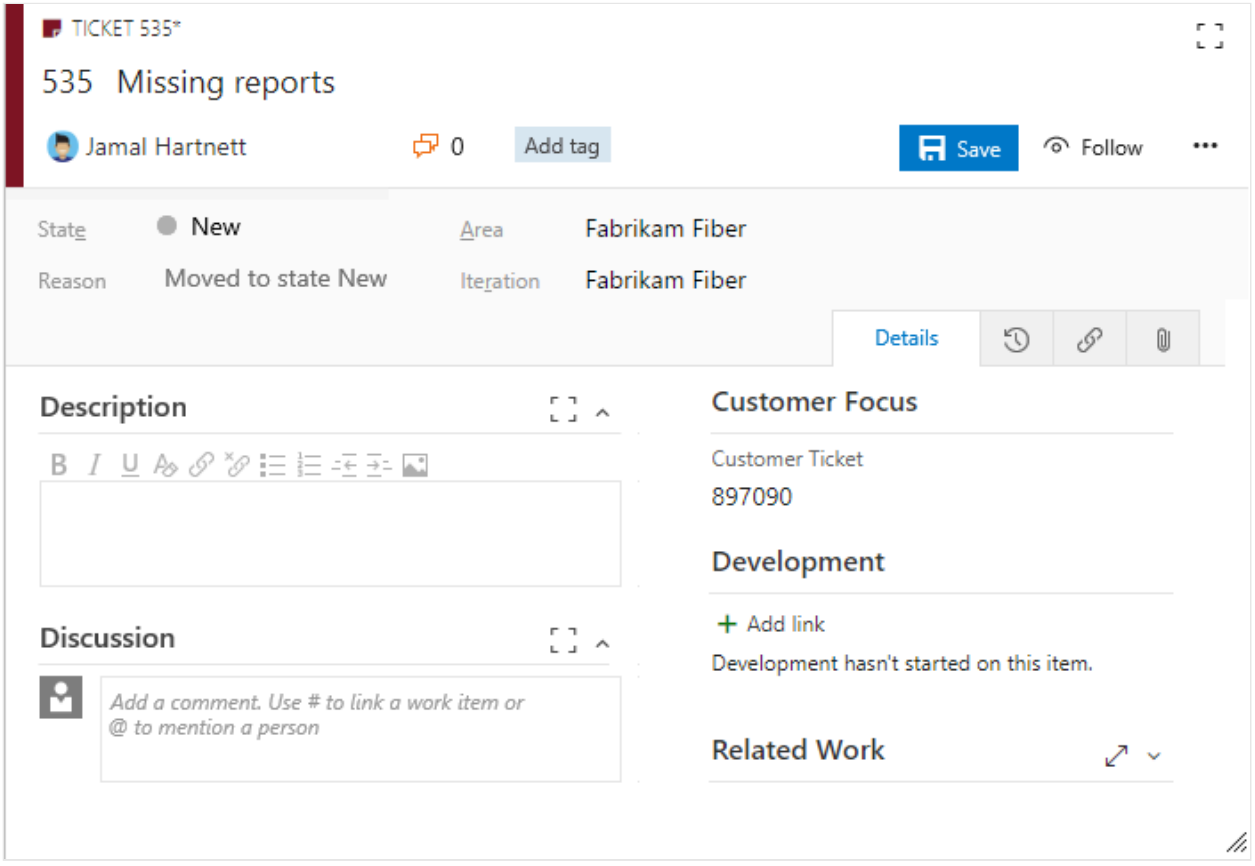
Add a custom work item type (Inheritance process)

Article • 07/24/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

You use different work item types (WITs) to plan and track different types of work. The main reason you add a custom WIT is to customize the web form and workflow states to meet specific business use cases. Or, you can customize an existing WIT. Your project contains 9 or more WITs that you can customize, based on the process used to create your project.

For example, you may want to capture customer issues in a custom WIT labeled Ticket.



To learn more about what you can customize, see About process customization and inherited processes.

> **Tip**
>
> To customize a single project, always start by creating an inherited process and migrating project(s) to that process. Then, all the customizations that you make to the inherited process automatically appear for the project you migrated.

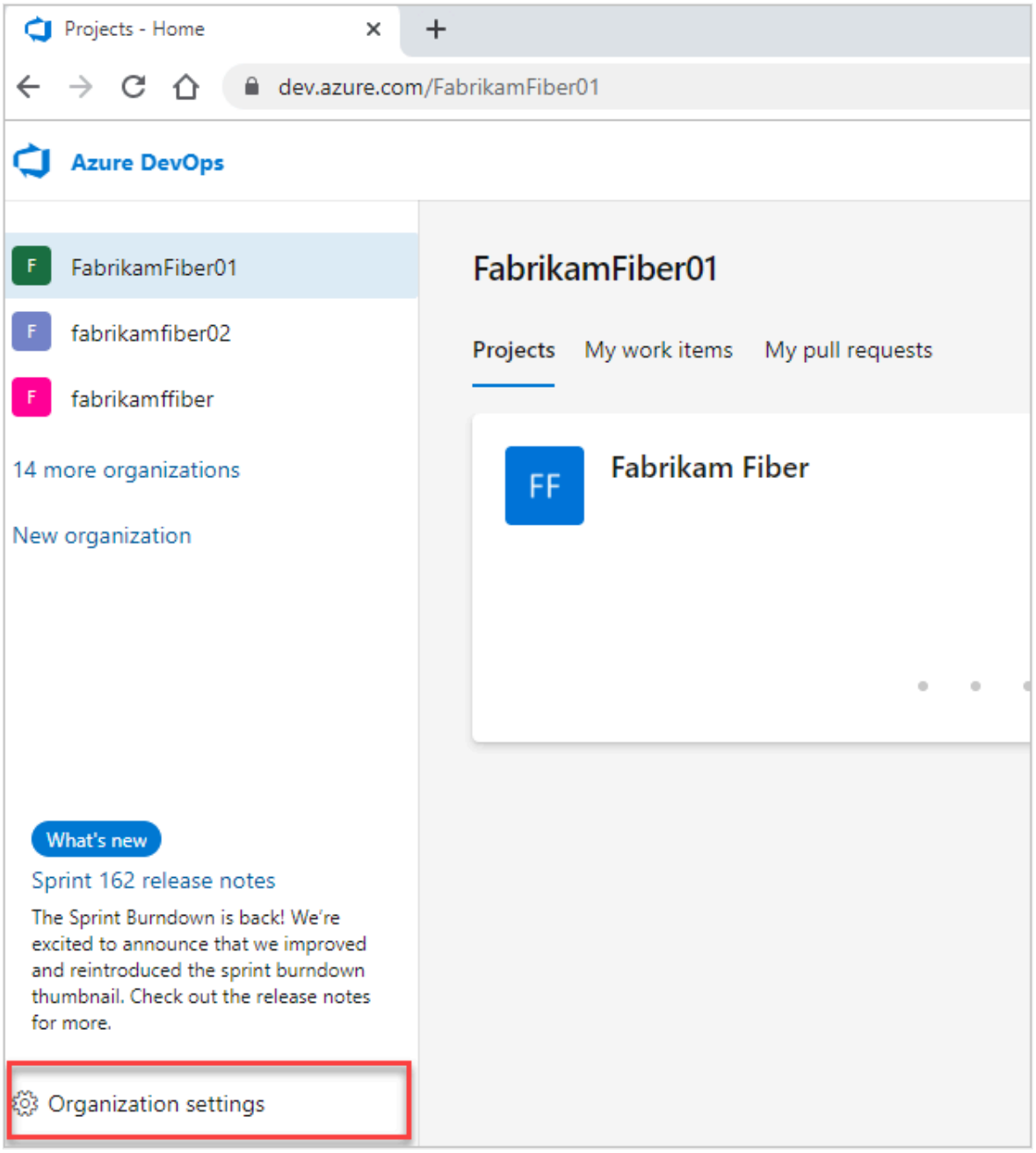
Prerequisites

Check out Configure and customize Azure Boards, which offers guidance on tailoring Azure Boards to align with your specific business requirements.

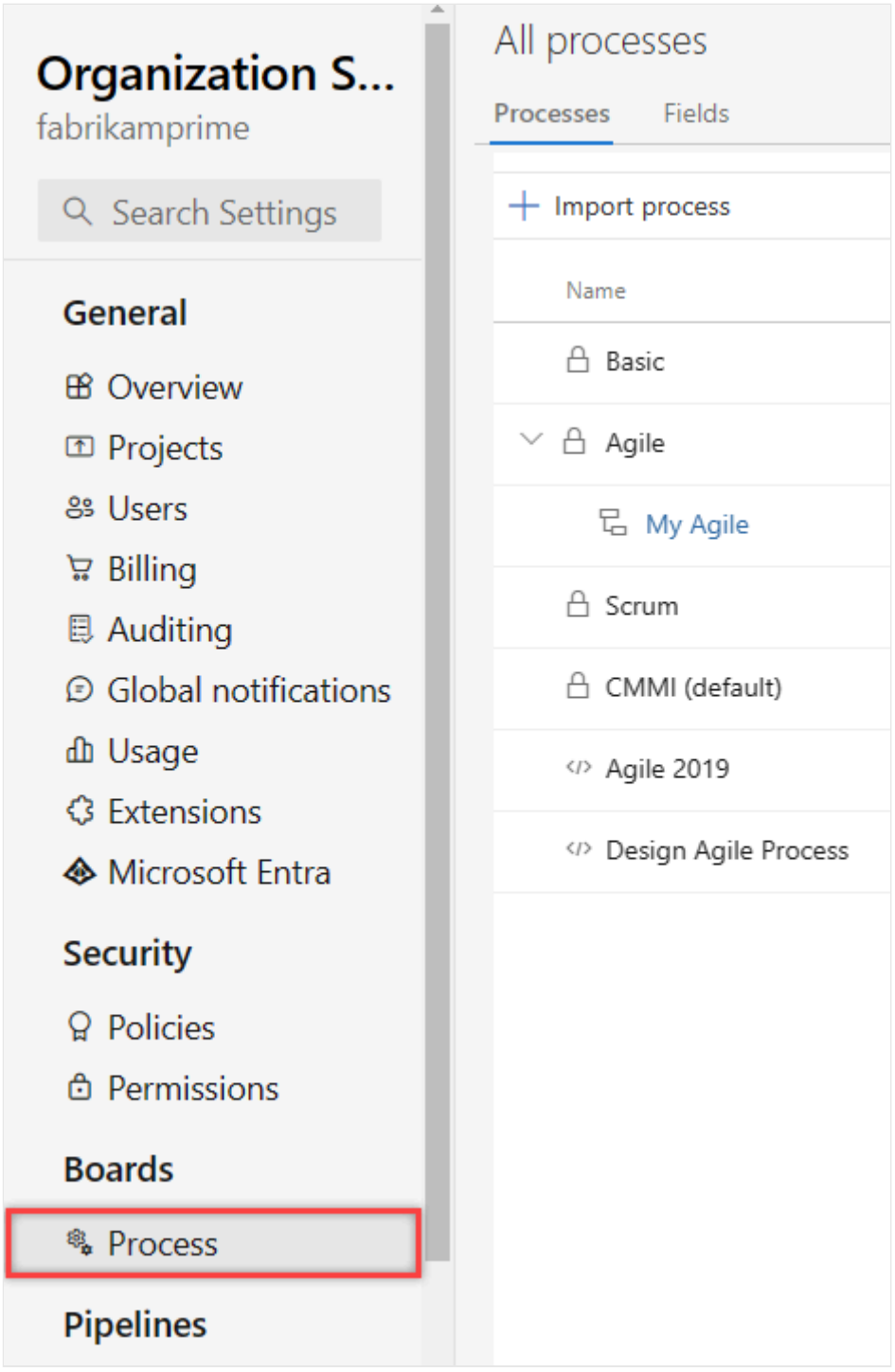
- **Organization requirement**: Ensure you have an organization in Azure DevOps.
- **Permissions**:
  - Be a member of the **Project Collection Administrators** group.
  - Have collection-level permissions such as **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**.
  - These permissions allow you to modify processes and fields within your organization.

Open organization process settings

1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`).
2. Select **Organization settings**.



3. Select **Process**.

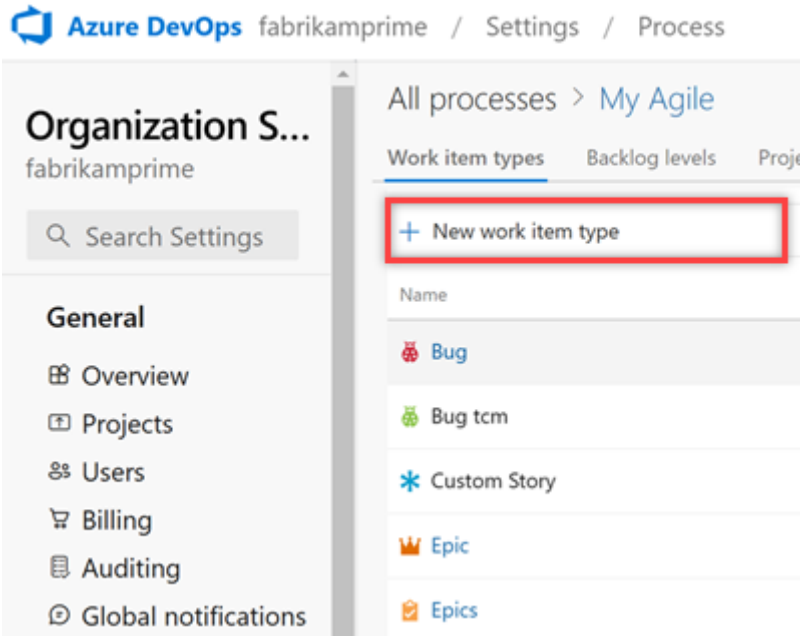


Create inherited process

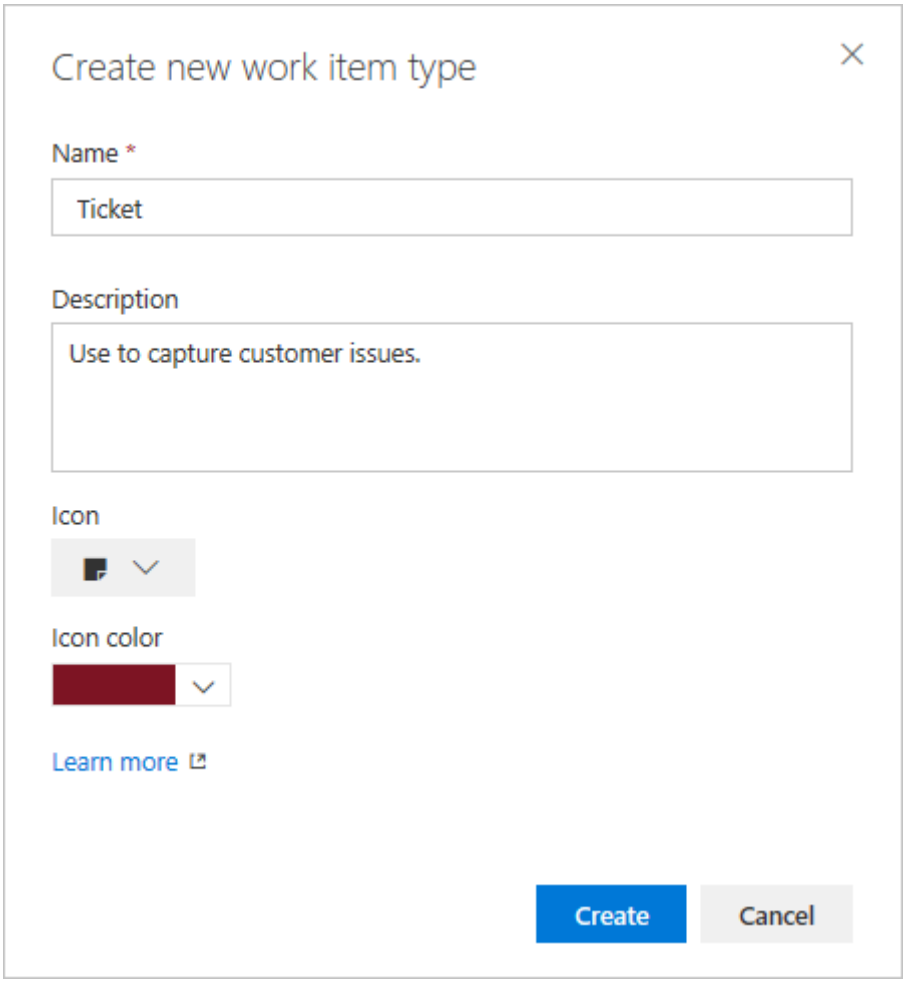
For more information, see Create inherited process.

Add a work item type

1. From the **Work Item Types** page, choose the + **New work item type**.

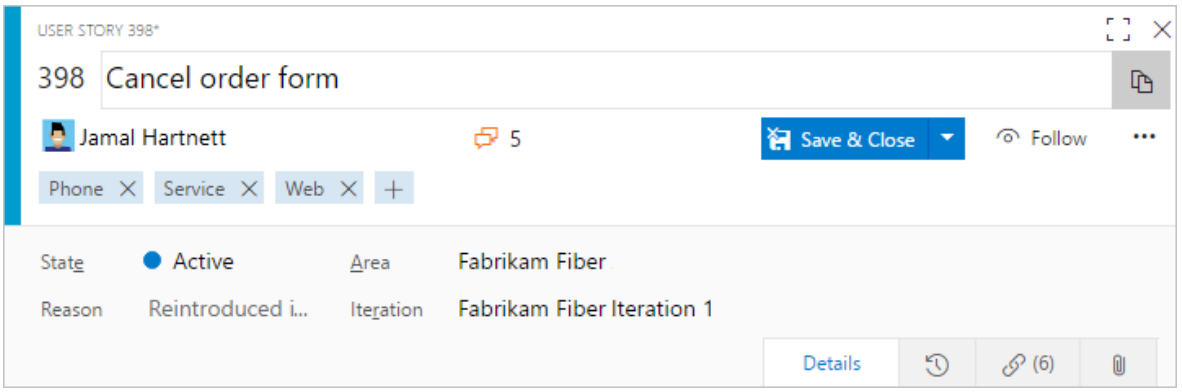


2. Name the WIT and optionally specify a description, icon and color. The icon and color you specify appear throughout the web portal, including on the work item form and when associated work items appear on a backlog, boards, query results, and more.



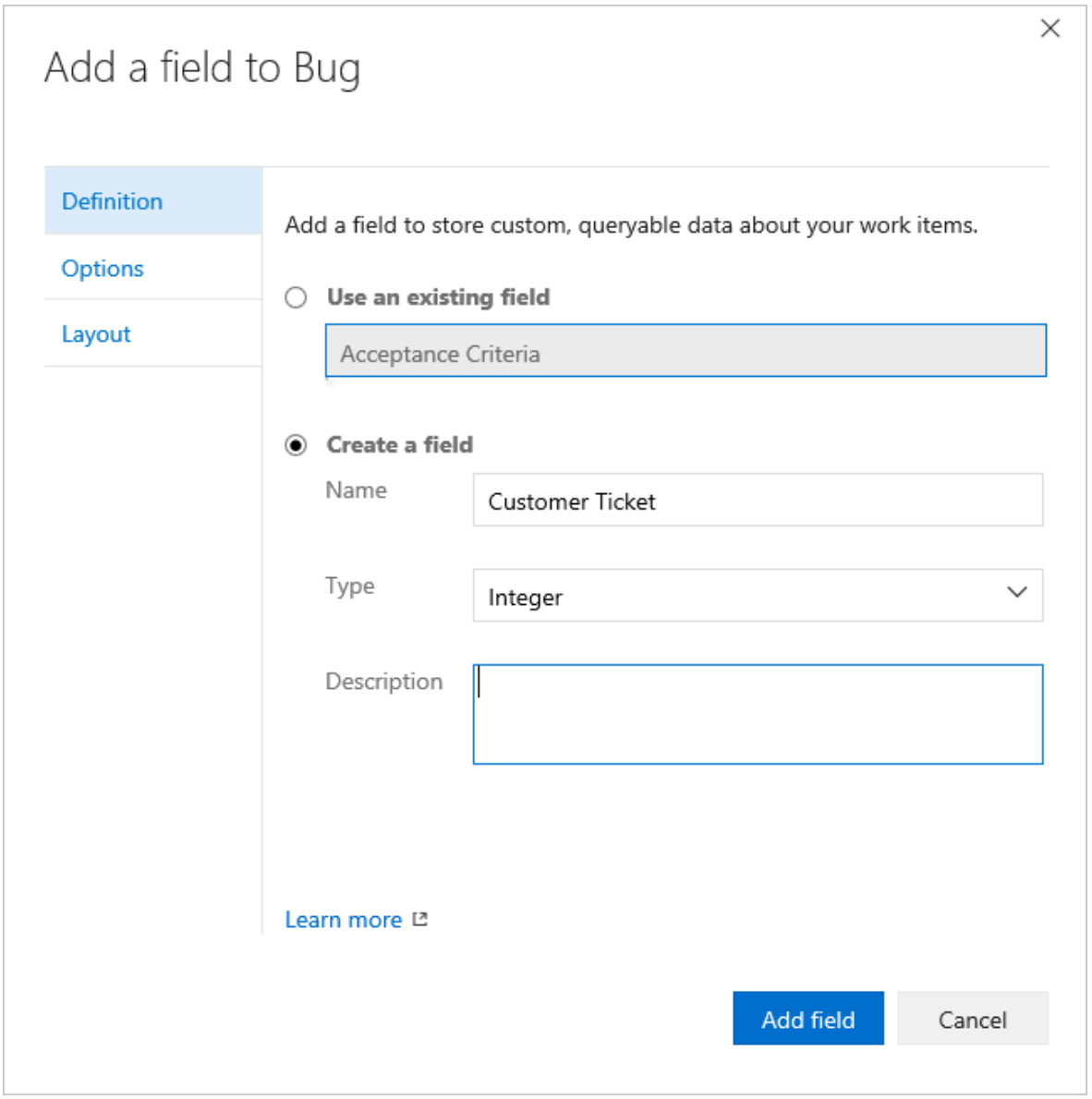
Choose **Create** to save.

Each new WIT comes predefined with a Details page with the Description field, and Discussion, Development, and Related Work groups. Also added, but not shown nor editable, are the standard elements included with the header of the form as shown in the following image, as well as the history, links, and attachment pages. For more information, see About work items.



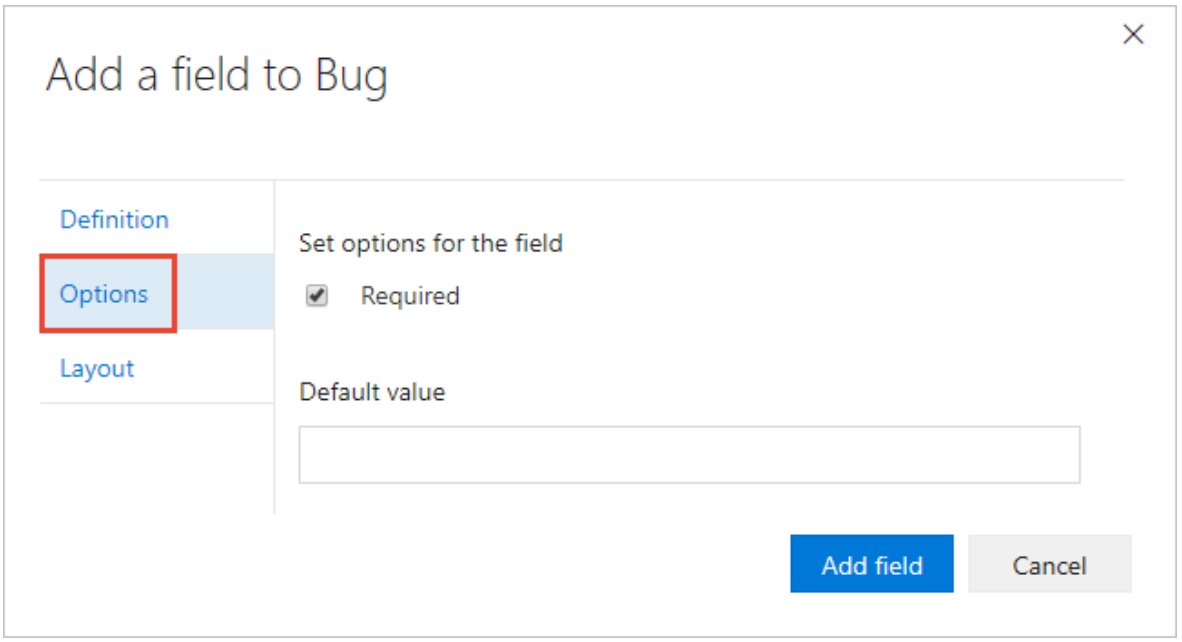
3. Name the field and select the field type from one of the supported data types. Field names must be unique and no more than 128 characters. For additional restrictions, see What is a field? How are field names used?. Optionally, add a description.

   Here we add an Integer field labeled Customer Ticket.



Additional data types you can add include: Picklist, Identity, Rich-text, HTML, and Checkbox.

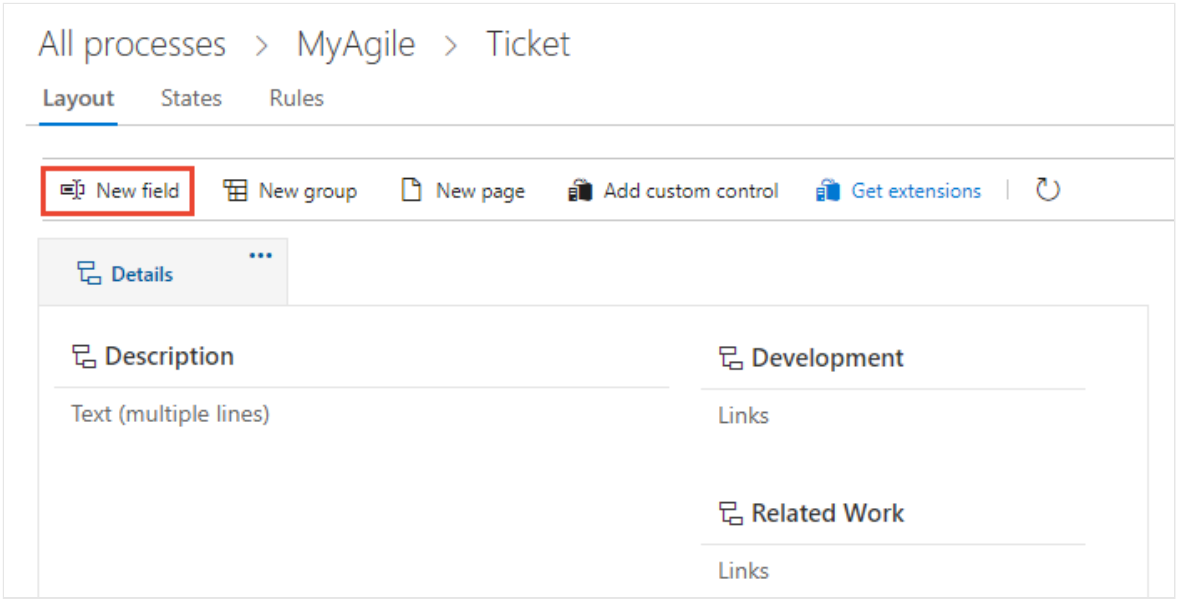
4. (Optional) On the **Options** tab, indicate if the field is required and specify a default value. Or leave these blank.



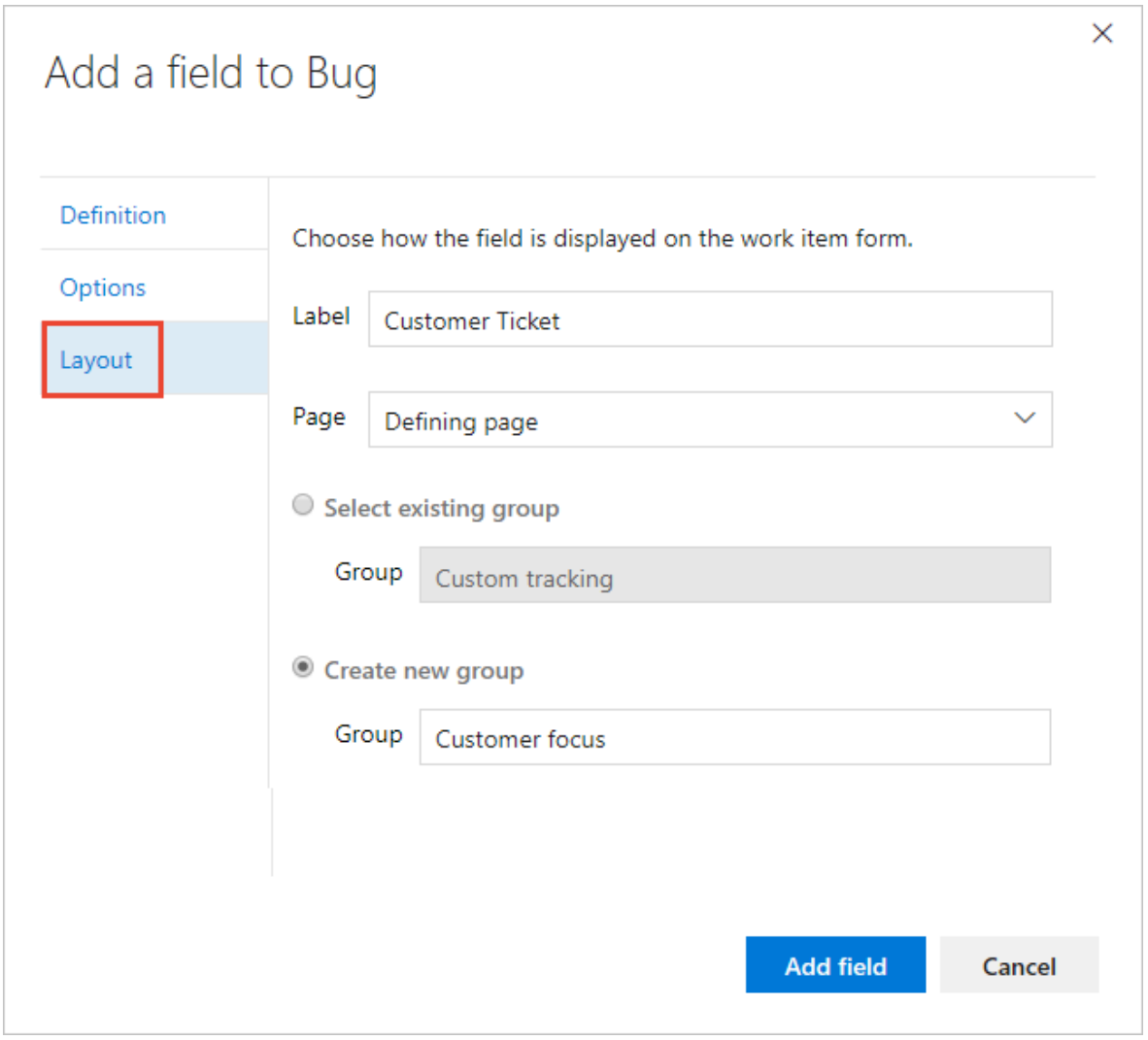
By making a field Required, users must specify a value for the field in order to save it. The default value you specify is set when you create a work item as well as every time a work item is opened and the field is empty.

5. (Optional) On the **Layout** tab, you can enter a different form label than the name of the field. Also, you can choose the page and group where the field appears on the form.

   Here we choose to add a new field. Choose the New field (**New Field** icon).



6. Here, we add the Customer Ticket field to a new group labeled Customer focus.



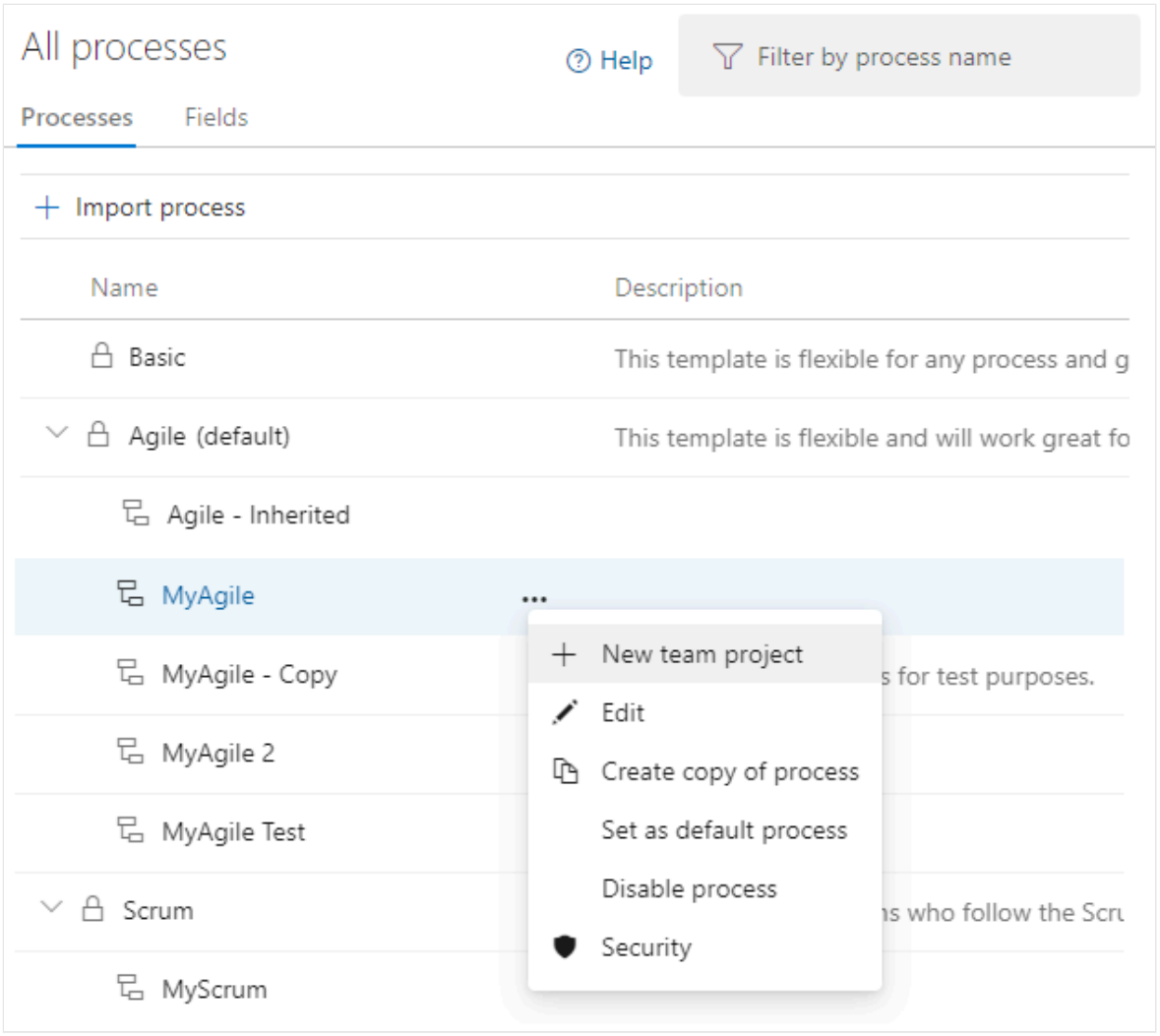
7. Choose **Add field** to complete adding the field. If you didn't specify its layout location, it gets added to the first group of fields on the layout form.

> **Tip**
>
> Once you've added a field, you can drag-and-drop it within a page to relocate it on the form. If you have several fields you want to add to a custom page or group, then you may want to add those elements first and then add your fields.

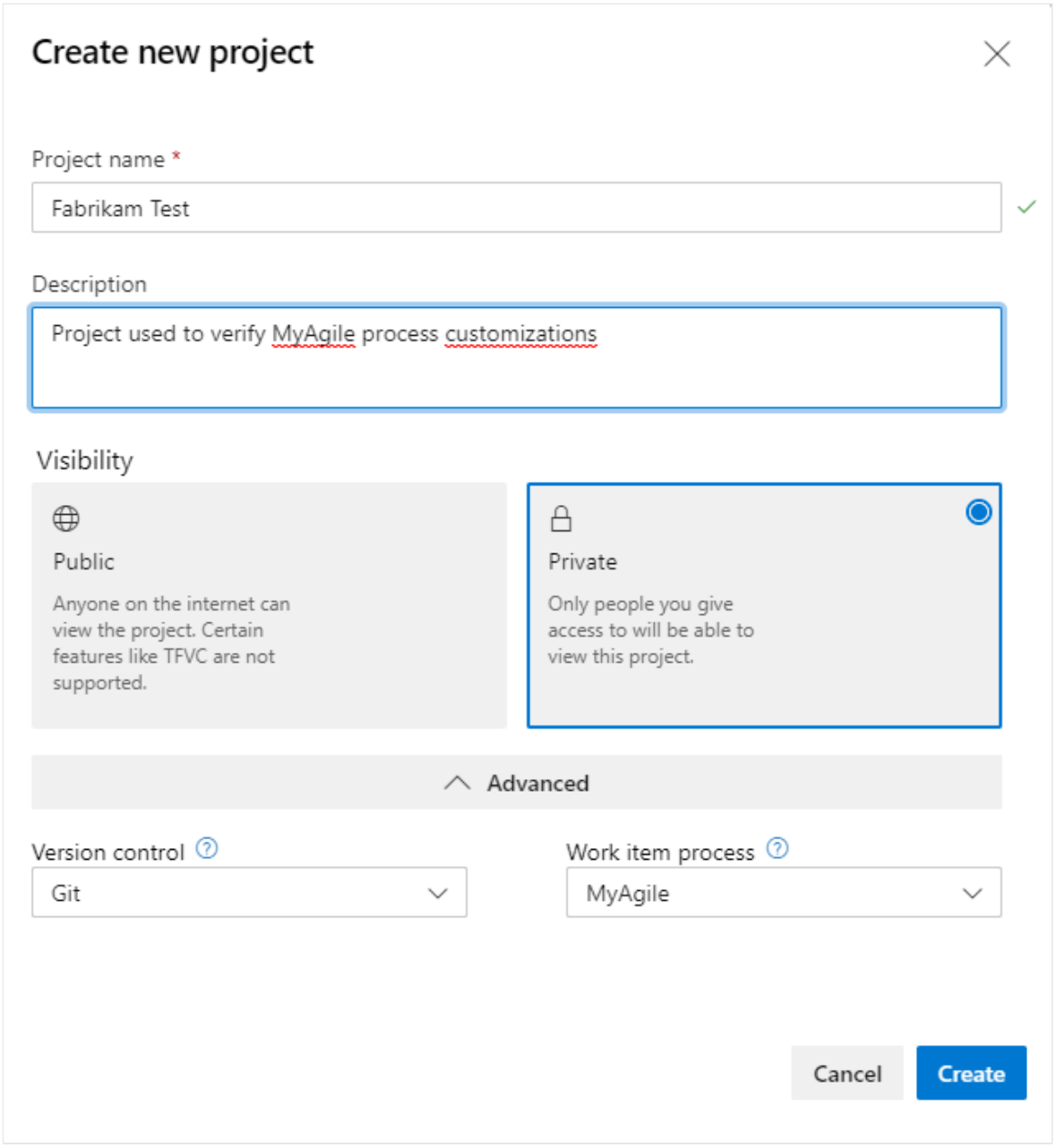
Verify the customization you made

We recommend that you create a test project and apply your customized inherited process to it to verify the changes you've made.

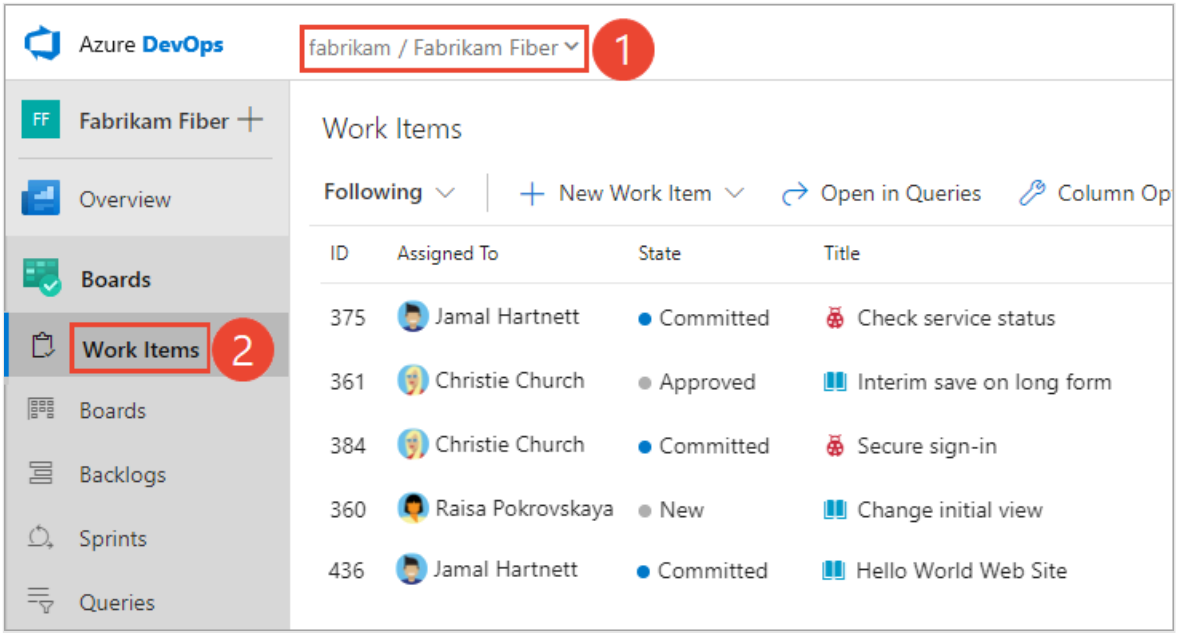
1. Open the **All processes** page, and choose the ... context menu for the process you want to use, and then select **New team project**.



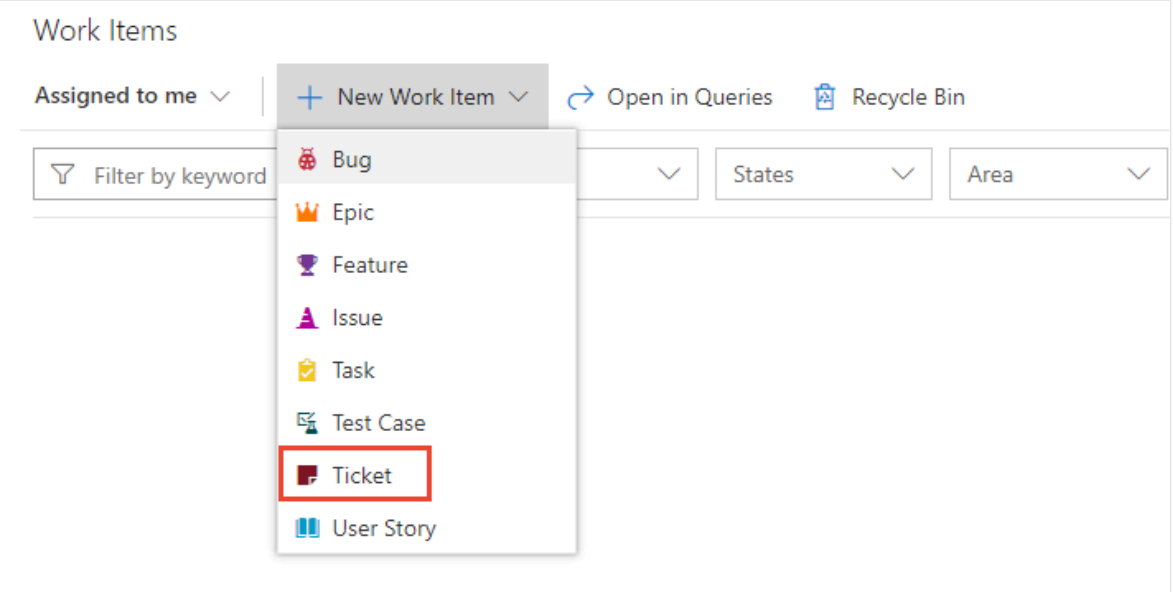
2. The Create new project page opens. Fill out the form.



3. Open **Work Items**. (1) Check that you have selected the right project, then (2) choose **Work>Work Items**.

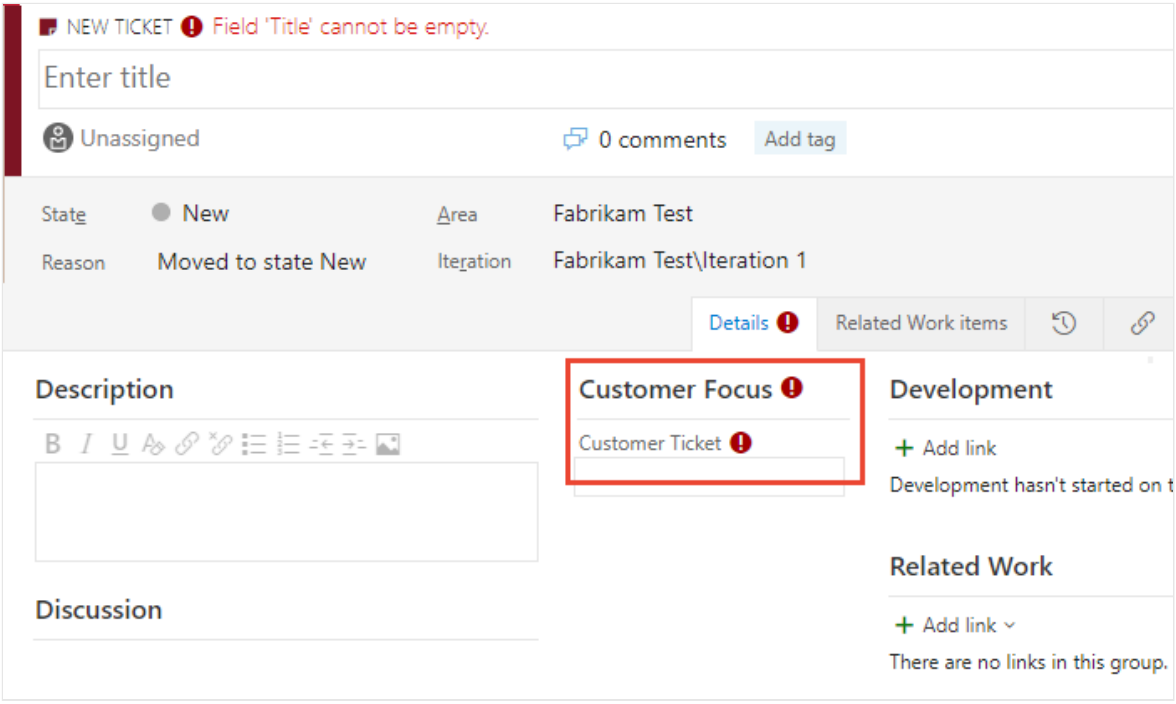


4. Select the WIT you customized. Here we choose **Ticket**.



   If you don't see the custom WIT, refresh your browser to make sure it registers all the custom changes you've made.

5. Verify that the field you added appears on the form. Note that the (exclamation mark) icon indicates the field is required.



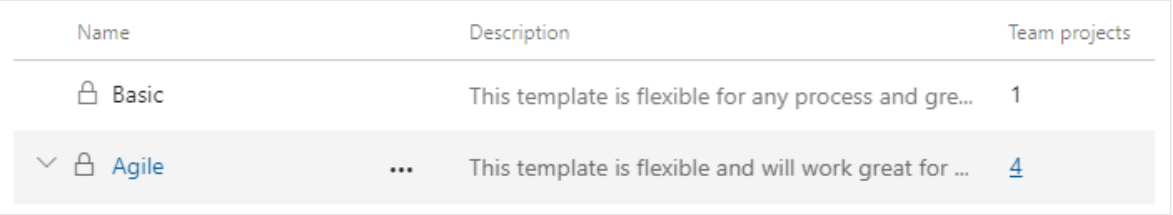
Apply the customized process to your project

After you've verified your customizations, you can now apply the process to your existing project.

> **Tip**
>
> As you customize a WIT, all projects that reference the inherited process that you're customizing automatically updates to reflect the custom WITs you've added. To view your customizations, refresh your web browser.

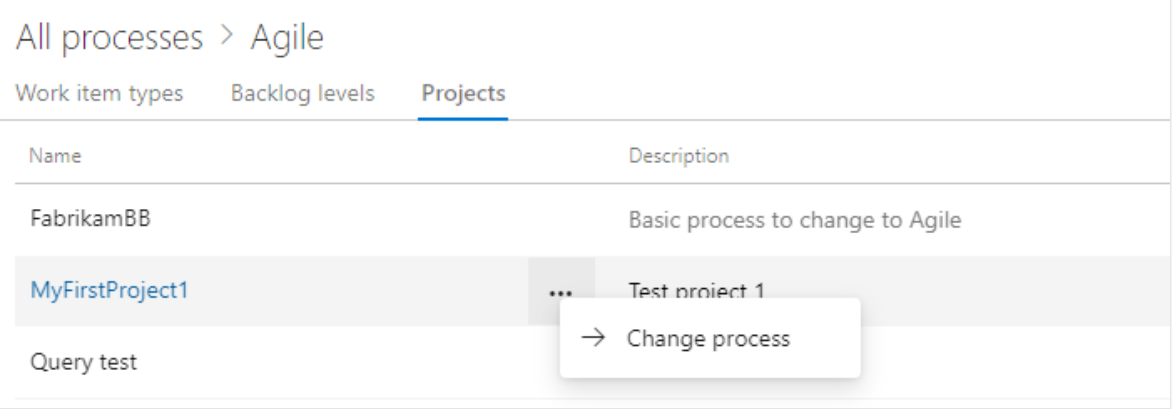
1. For the process currently used by the project, choose the number of projects.

   Here we open the menu for the Agile default process.

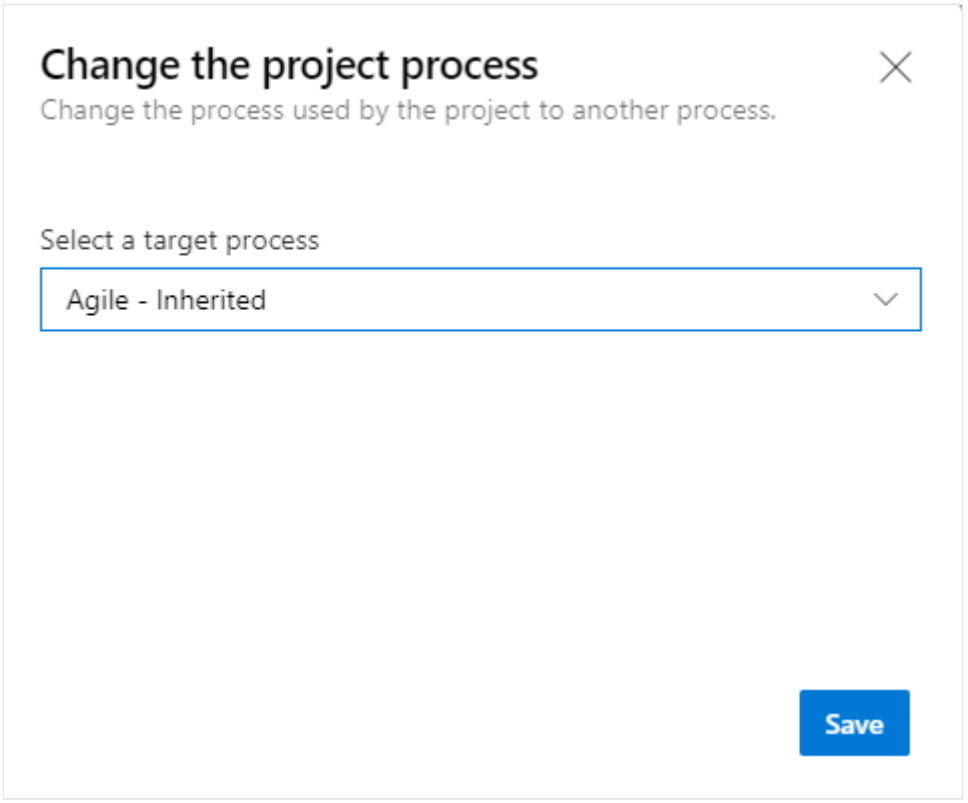
   

2. Open the … context menu for the project you want to change, and choose the **Change process** option.

   Here we open the menu for the MyFirstProject1.

3. From the Change the project process dialog, choose the process from the menu of options. And, then choose **Save**.



Q & A

Q: How do I get my custom work item type to show up on my backlog?

A: Modify your requirement backlog to include the custom work item type. For details, see Customize your backlogs or boards, Edit or rename the requirement backlog.

> **Note**
>
> The backlog level to which you add a custom work item type determines the parent work item type(s) for the work item type.

Try this next

Customize the web layout Or Customize the workflow

Related articles

> **Note**

Review changes made to an inherited process through the audit log. For more information, see Access, export, and filter audit logs.

- Customize a project using an inherited process
- Create and manage inherited processes
- Customize the workflow states.
- Customize your backlogs or boards for a process.

Feedback

Was this page helpful? Yes No

Provide product feedback

Customize a project using an inherited process

Article • 08/15/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Each project is based on a process that defines the building blocks for tracking work. The first project you create uses one of the default processes—**Agile**, **Basic**, **Scrum**, or **CMMI**.

You can only customize inherited processes. Any changes you make to the inherited process automatically appear in the projects that use that process. You can quickly customize a process by adding or modifying a work item type (WIT) that's part of the process. Otherwise, use the following sequence to customize a project:

1. **Customize an inherited process:** Modify the inherited process to align with your specific requirements by modifying fields, WITs, workflows, forms, and backlogs.
2. **Verify your customizations:** Create a test project and validate your changes.
3. **Apply inherited process to a project:** Add a new project based on inherited process and change the process used by an existing team project.
4. **Refresh and verify your customizations:** Refresh the web portal and open a work item of the type you modified.

> **Note**
>
> Review changes made to an inherited process through the audit log. For more information, see Access, export, and filter audit logs.

Prerequisites

Check out Configure and customize Azure Boards, which offers guidance on tailoring Azure Boards to align with your specific business requirements.

- **Organization requirement**: Ensure you have an organization in Azure DevOps.
- **Permissions**:
  - Be a member of the **Project Collection Administrators** group.
  - Have collection-level permissions such as **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**.

    - These permissions allow you to modify processes and fields within your organization.

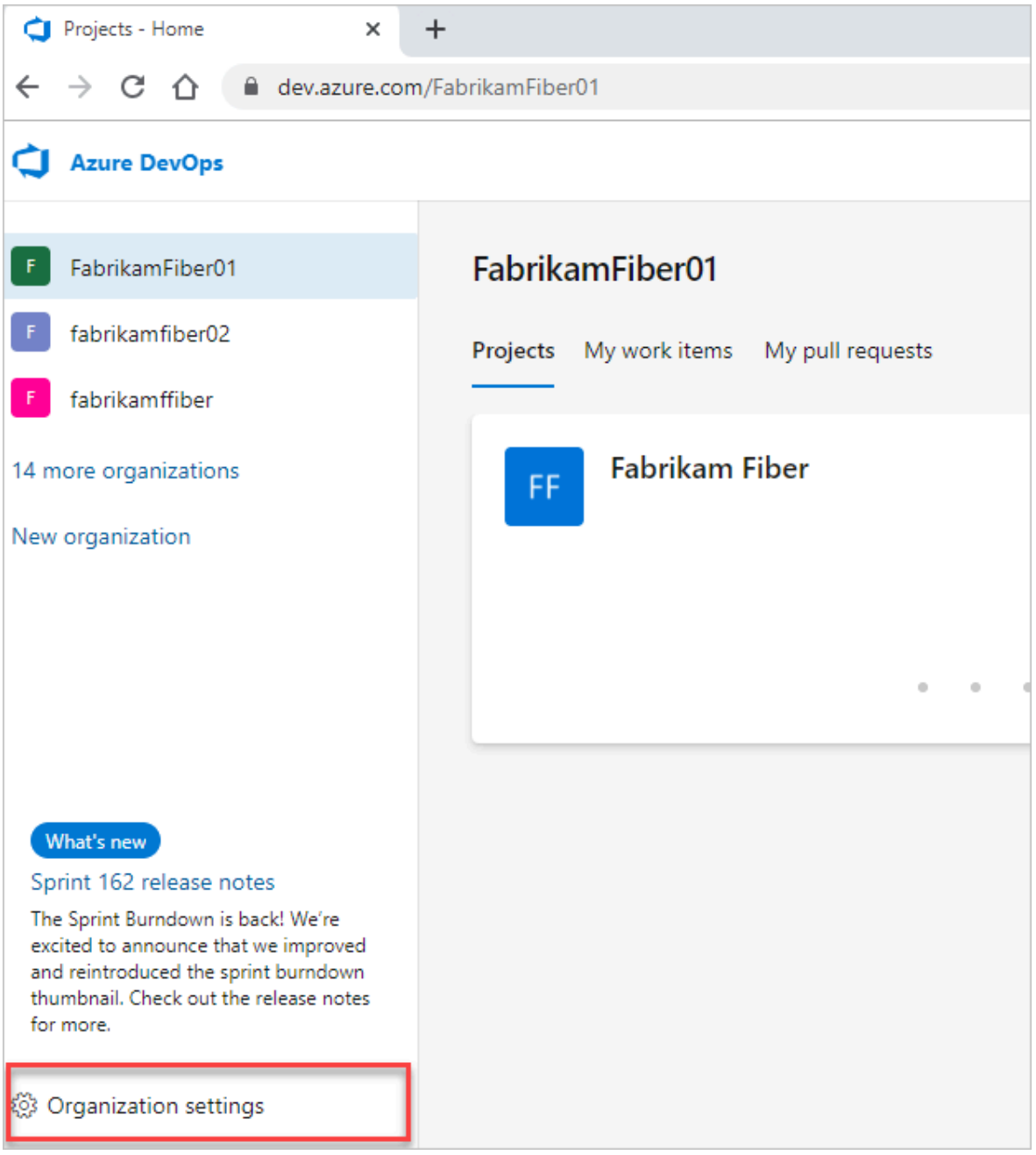
- **Project requirement:** Ensure that you have a project that uses an inherited process. For more information, see Create an inherited process.

> **Note**
>
> When you customize an inherited process, any projects using that process automatically reflect the customizations. To ensure a smooth transition, we recommend creating a test process and project, which allows you to test your customizations before you implement them organization-wide. For more information, see Create and manage inherited processes.

Add or modify a field

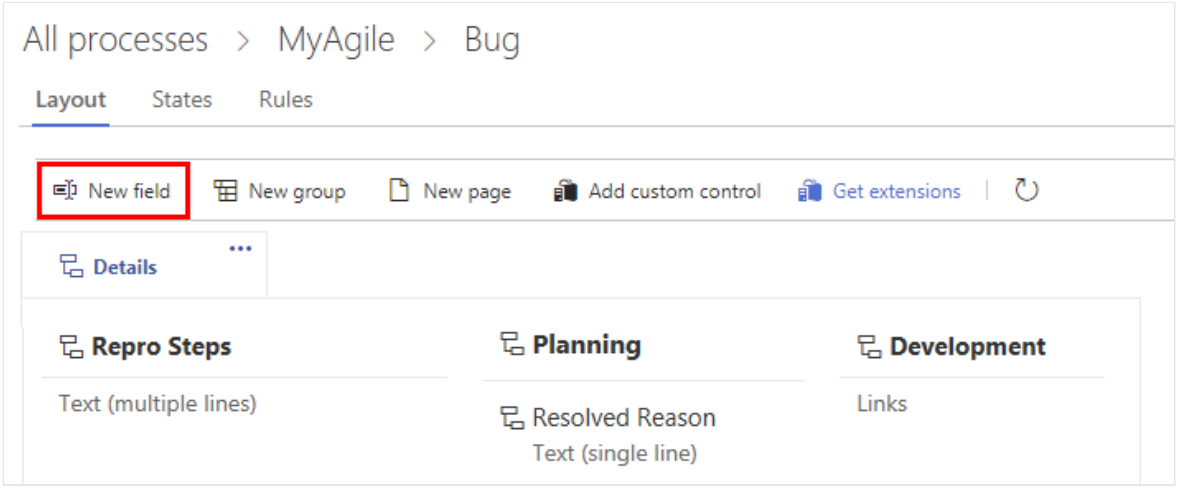
Locked fields and inherited fields correspond to inherited fields from a system process. You can't customize locked fields, but you can customize some options for inherited fields. You can fully customize fields that you add to a process.

1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`).
2. Select **Organization settings**.



3. Select **Process** > your **inherited process** > the **WIT** you want to customize.

4. To add a field, choose the New field (**New Field** icon).



In the resulting dialog, choose the type of field that you want to add. For example: integer, picklist (drop-down menu), person-name/Identity, rich-text or HTML, or checkbox (boolean).

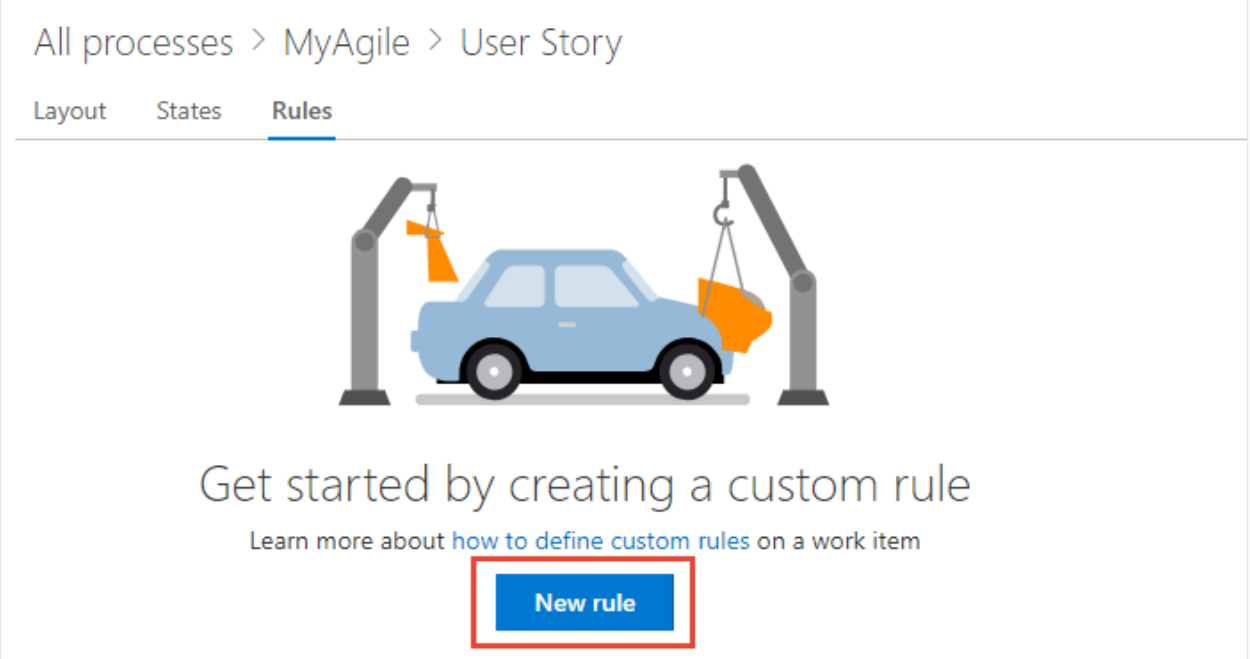
5. Modify an existing field in the following ways:

   - Change the field label
   - Show/Hide field on form
   - Set Required/Default options
   - Move the field within the layout
   - Remove field from the form

Add or modify a rule for a work item type

Add rules to support specific workflow and business use cases. Rules let you clear the value of a field, copy a value into a field, and apply values based on dependencies between different fields' values.

1. Select your inherited process and the work item type.
2. Select **Rules** > **New rule**.



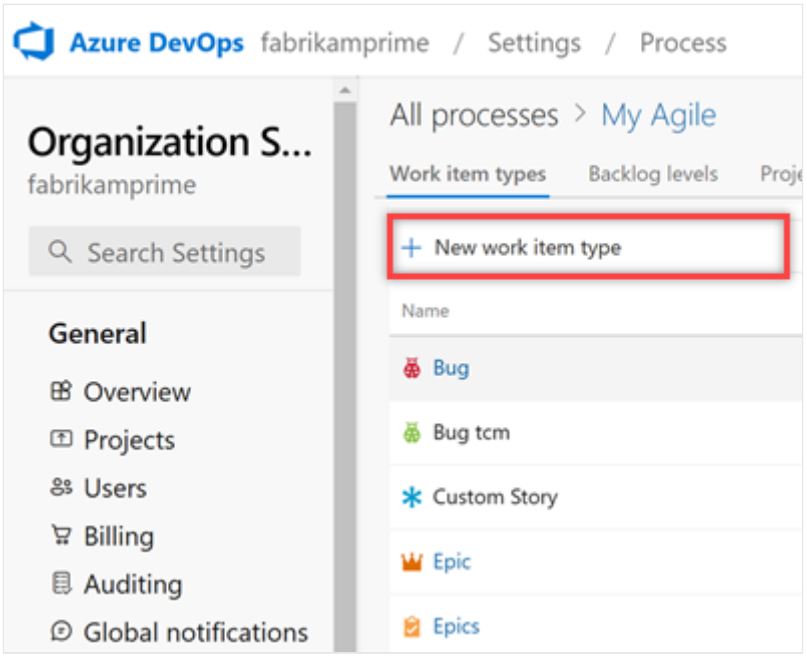
For more information, see Rules and rule evaluation.

Add or modify work item types

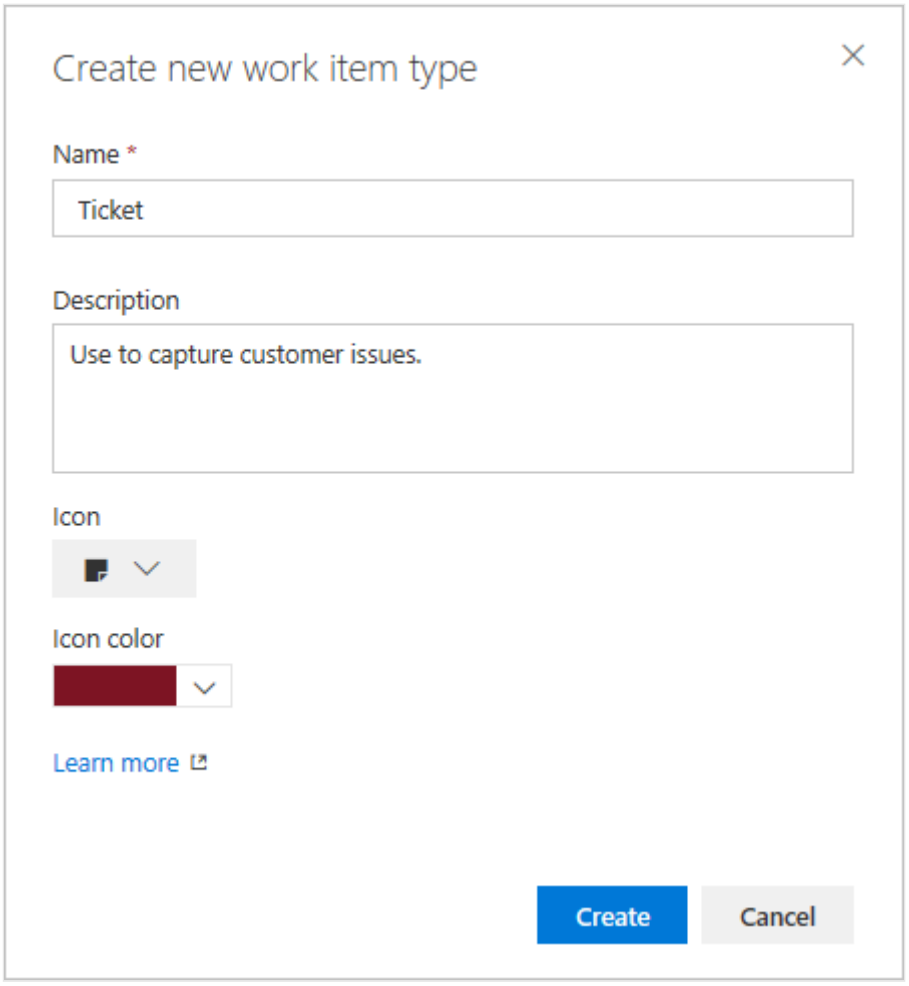
You use different WITs to plan and track different types of work. The main reason you add a custom WIT is to customize the web form and workflow states to meet specific

business use cases.

1. Select your inherited process and the WIT you want to customize.
2. From the **Work Item Types** page, choose the + **New work item type**.



3. Name the WIT and optionally specify a description, icon, and color. The icon and color you specify appear throughout the web portal, including on the work item form and when associated work items appear on a backlog, boards, query results, and more.



4. Select **Create** to save.

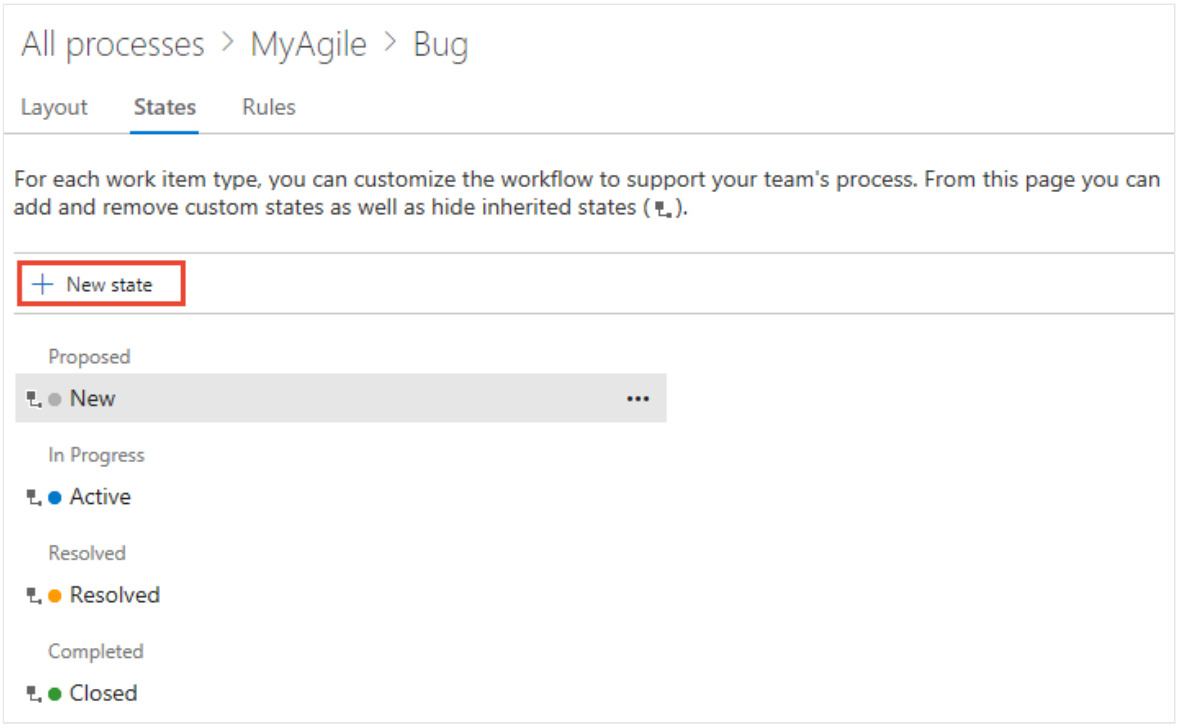
   You can now add fields to the WIT, or customize it in the following ways:

   - Modify the workflow
   - Add or remove a custom control
   - Add an extension
   - Add/move/remove custom groups
   - Add/move/delete custom pages

Modify the workflow of a work item type

Workflow states allow you to track the status of a work item as it moves from new to completed.

1. To modify a workflow, choose your inherited process, the WIT, and then the **States** page.



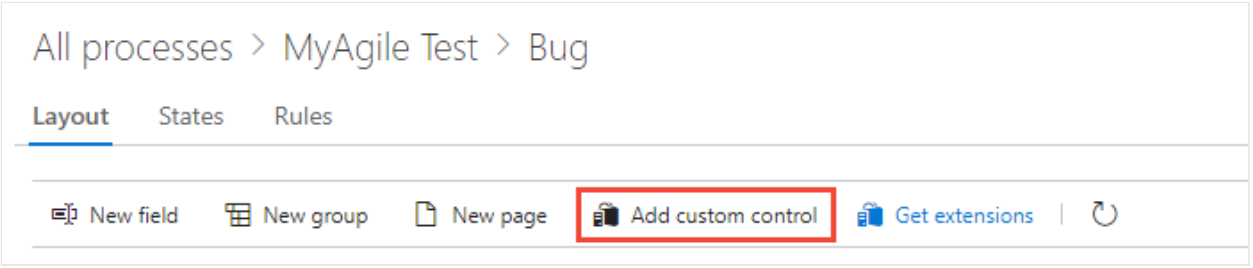
2. You can modify the workflow in the following ways:

   - Hide a workflow state
   - Add a state
   - Edit a state (change color or category)
   - Remove a state

Add a custom control

Custom controls provide more functionality to a work item form.

From the **Process** page, select your inherited process > WIT > **Add custom control**.



For more information, see Add extensions and custom controls to a work item type.

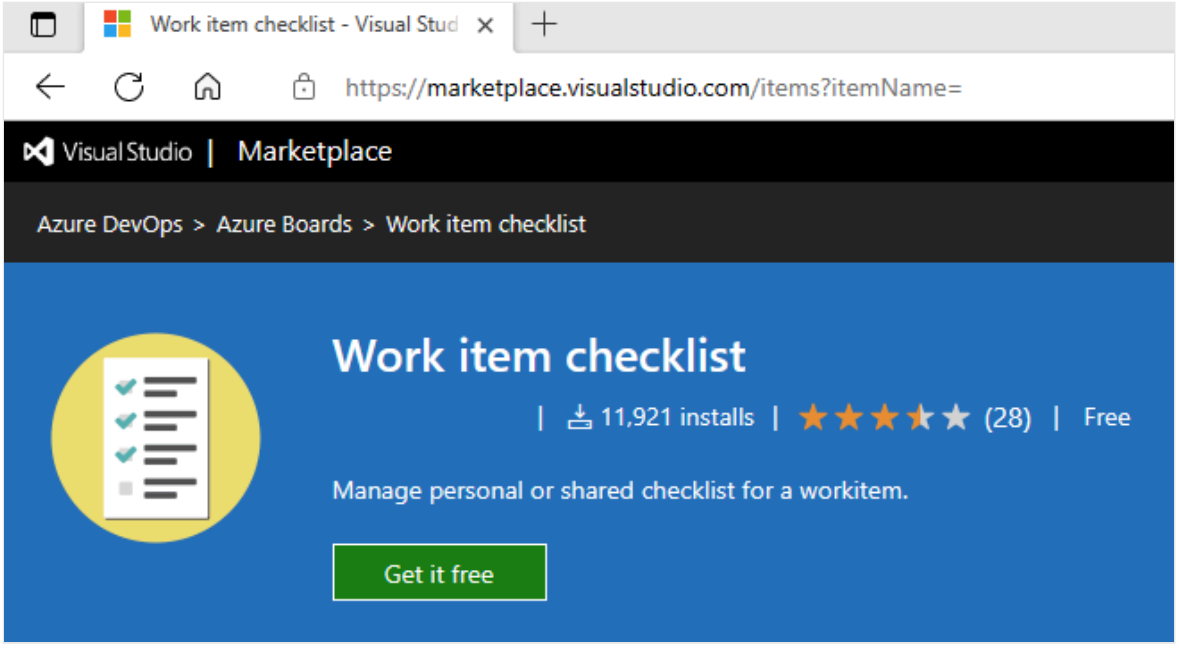
Add an extension to a work item type

An extension is an installable unit that contributes new capabilities to your project.

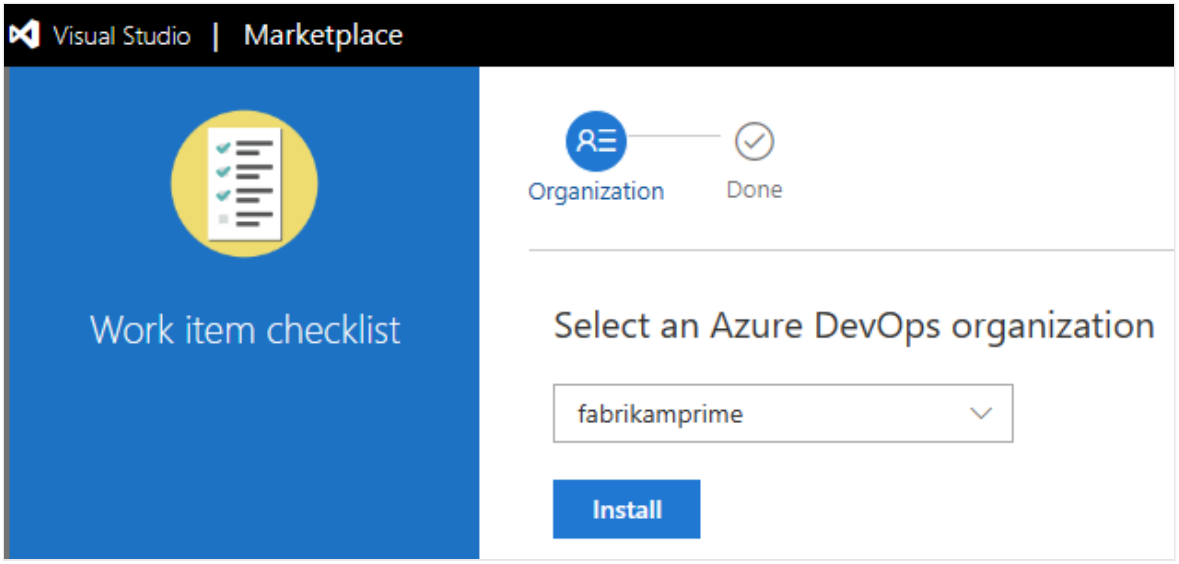
> **Note**

Group and Page extensions automatically get added to all WITs for all processes, both system and inherited. You can hide an extension for selected WITs within an inherited process.

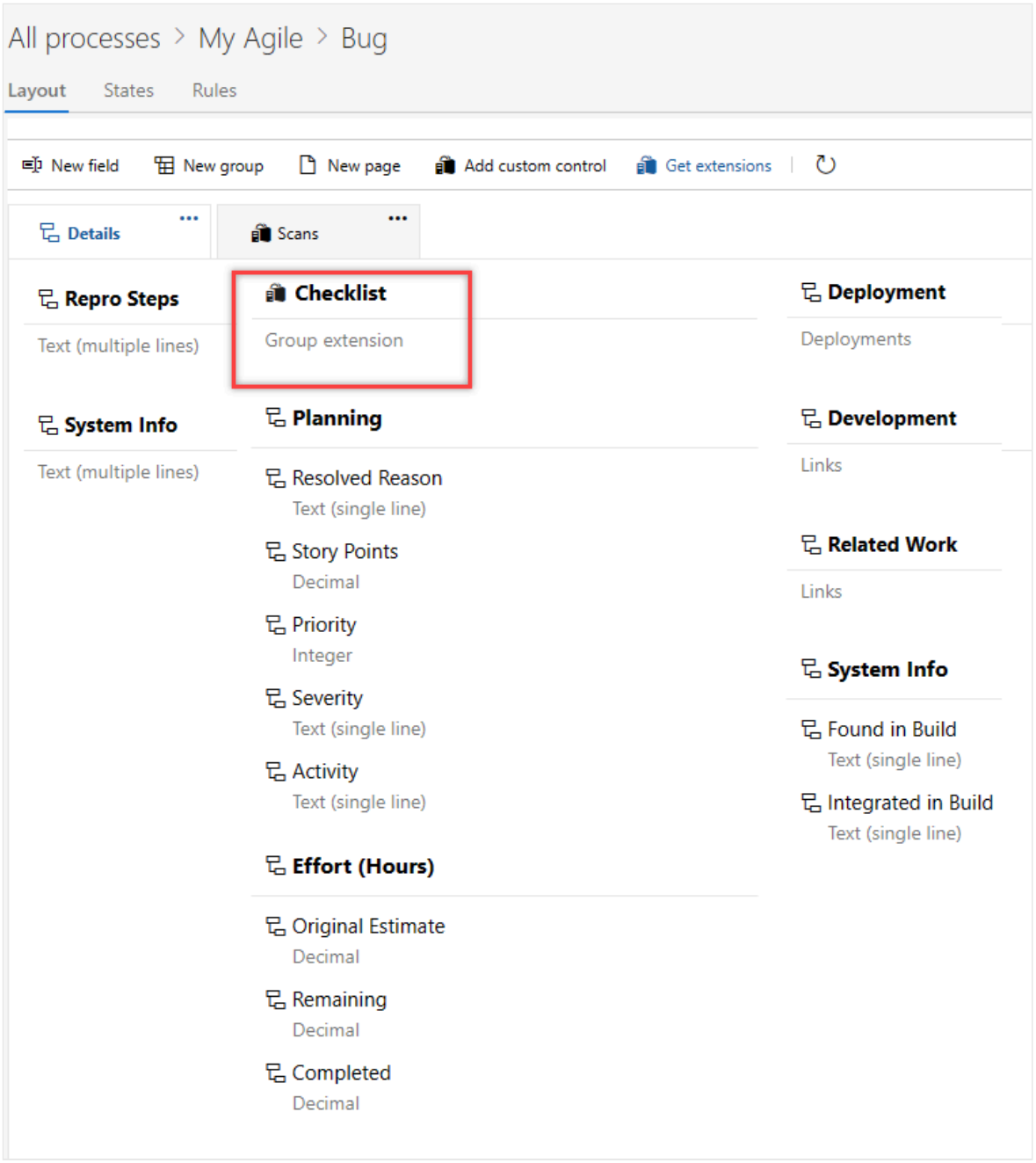
1. Go to the Visual Studio Marketplace, find an extension, and select **Get it free**.



2. Choose the organization you want to add it to from the dropdown menu, and then select **Install**.



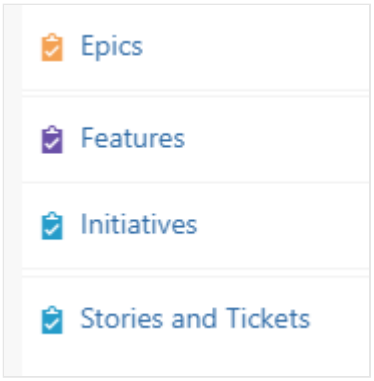
3. Return to the process and WIT and verify the extension is where you want it. You can drag it to where you want it on the form.



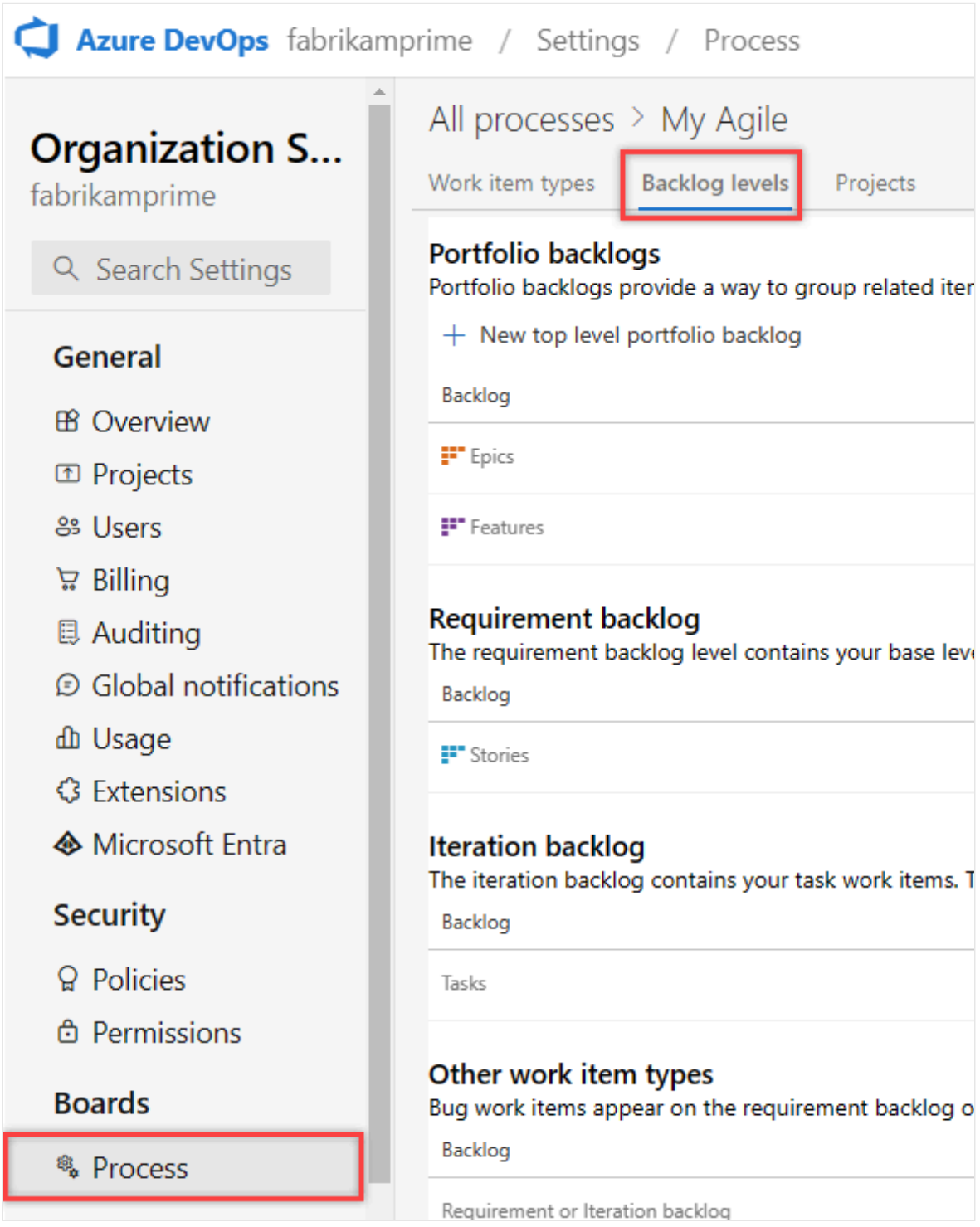
Modify the backlog and boards

You can add more WITs to a backlog level or create another portfolio backlog. For instance, consider the following example:

- We introduced a third-level portfolio backlog called **Initiatives**, specifically designed to track the custom **Initiative** WIT.
- We also renamed the product backlog to **Stories** and **Tickets**, signifying that it now encompasses both **User stories** and **Customer tickets**.



1. From the **Process** page, select your inherited process > **Backlog levels**.



2. You can modify the backlog and board configuration in the following ways:

- Add a custom WIT to a backlog or board
- Change the default WIT
- Rename the requirement backlog
- Rename a portfolio backlog
- Add a portfolio backlog that displays custom WITs
- Edit or rename a portfolio backlog
- Delete the top-level custom portfolio backlog

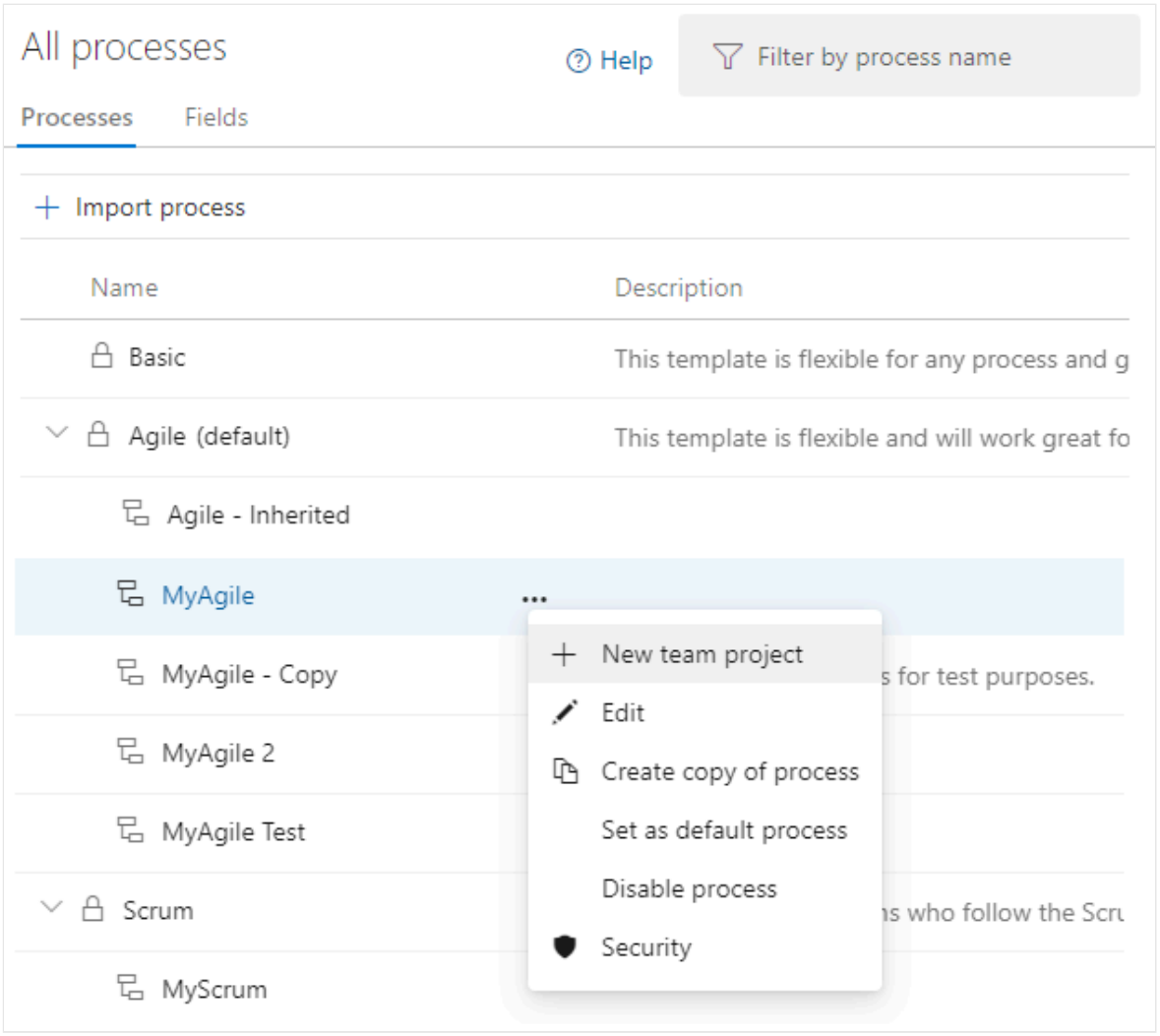
Verify your customization

We recommend that you create a test project and apply your customized inherited process to it to verify the changes you made. All customizations made to a process go into effect immediately on all projects. If you want to stage your changes, you can do that using one of two methods:

- Create a test project and copy of your customized process
- Create a test organization and import/export your process

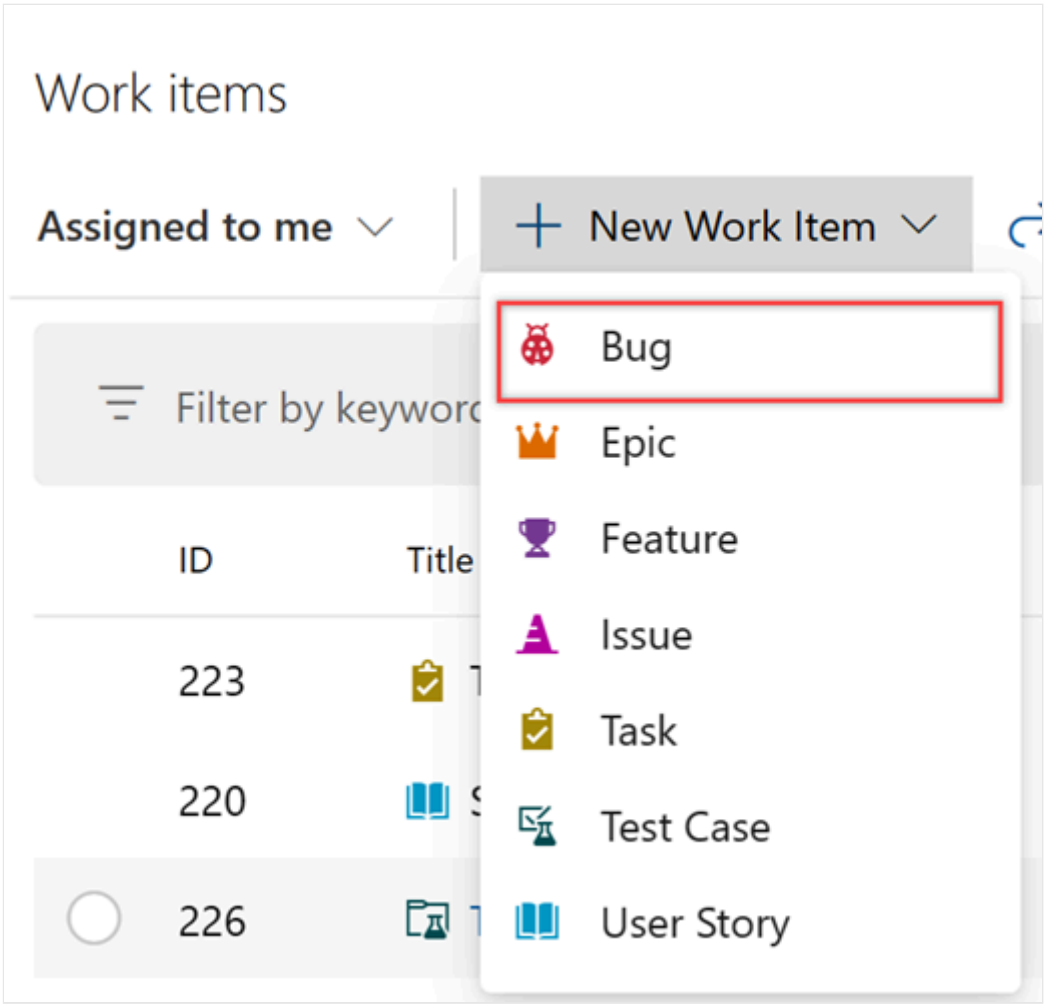
Create a test project and copy your customized process

1. From the **Process** page, select the … context menu for the process you want to use, and then select **New team project**.



2. Enter information into the form, and then select **Create**. For more information, see Create a project.

3. From your project, select **Boards** > **Work Items**, and then select the customized WIT from the **New Work Item** dropdown menu. In the following example, we select **Bug**.



4. Verify that one or more fields you added appear on the form. The (exclamation mark) icon indicates the field is required.

Create a test organization and import/export your process

You can also use the following steps to verify the customizations you made to an inherited process.

1. Create a test organization.
2. Use the import/export process tool to copy the process to the test organization.
3. Verify the process customizations in the test organization.
4. Use the import/export process tool again to import the modified process to the production organization.

Change your project's process

For more information, see Change a project's process.

Related articles

- Use the REST API to manage processes
- Create an inherited process and change the project to use that process.
- Learn work tracking object limits

Feedback

Was this page helpful? Yes No

Provide product feedback

Create and manage inherited processes

Article • 06/04/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

In Azure DevOps, you have the flexibility to customize your project, Agile tools, and the work tracking system by using inherited processes. These customizations apply to all projects that utilize the same process.

An inherited process serves as the foundation for your work tracking system. When you create a new project, you choose a process to define its building blocks. These building blocks include work item types, states, fields, and rules. By customizing an inherited process, you tailor it to your team's specific needs.

For more information about what you can customize, see About process customization and inherited processes.

> **Note**
>
> Review changes made to an inherited process through the audit log. For more information, see Access, export, and filter audit logs.

Prerequisites

We recommend checking out Configure and customize Azure Boards, which offers guidance on tailoring Azure Boards to align with your specific business requirements.

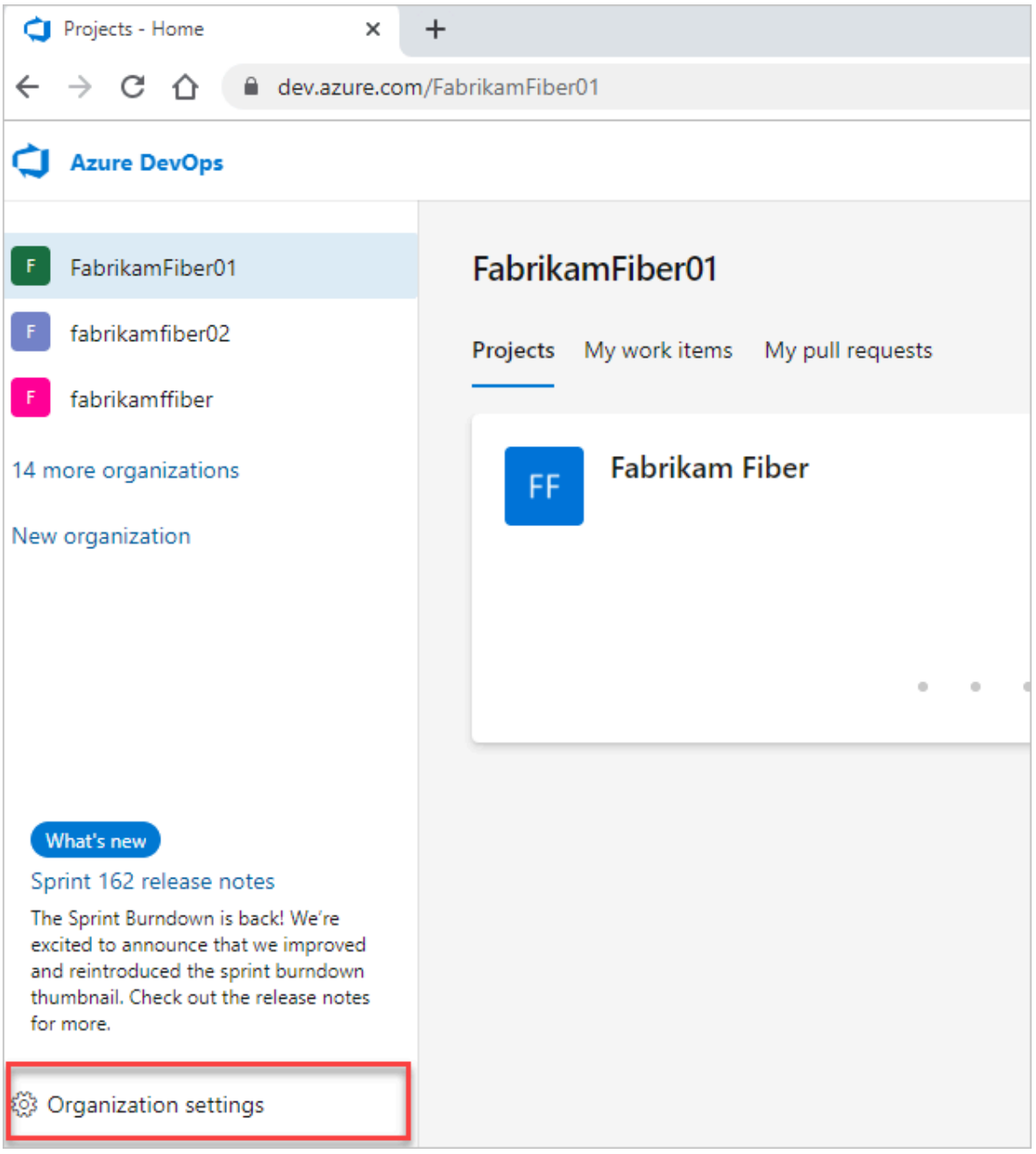
- You must have an organization in Azure DevOps.
- To create, edit, and manage processes, you must be a member of the **Project Collection Administrators** group, or have the corresponding collection-level permissions **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**. For more information, see Set permissions and access for work tracking, Customize an inherited process.

Create an inherited process

Do the following steps to create an inherited process that you can customize. The default, system processes are locked, so you can't customize them.

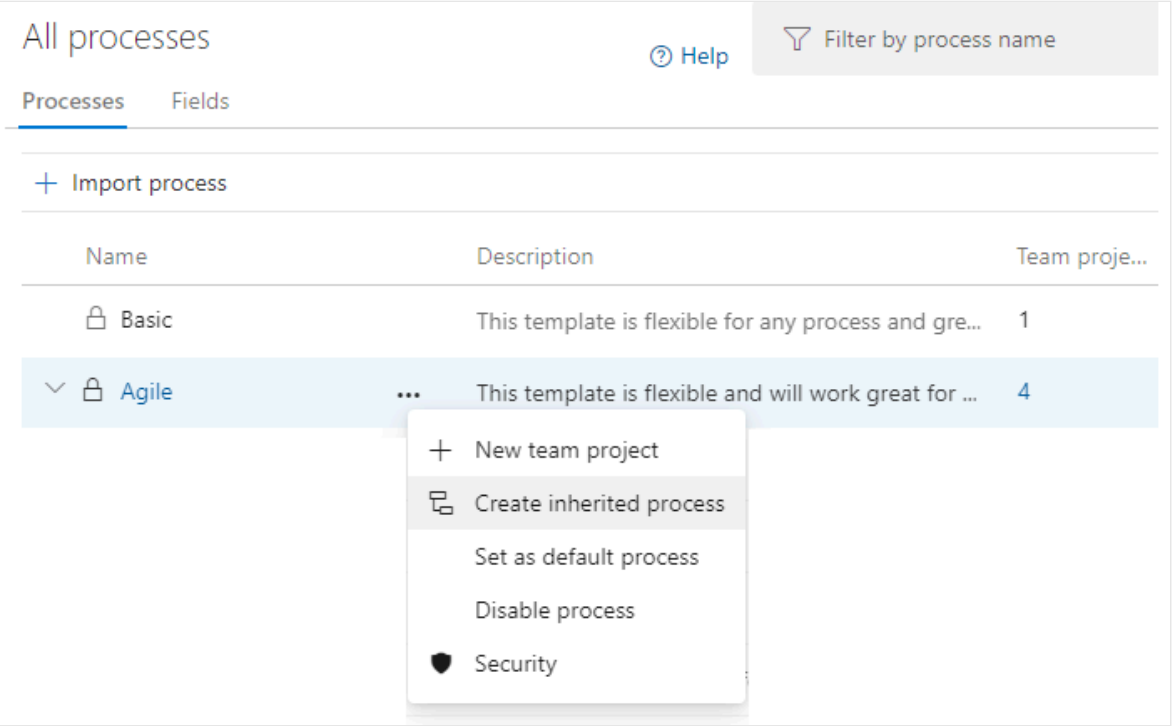
1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`).

2. Select **Organization settings**.



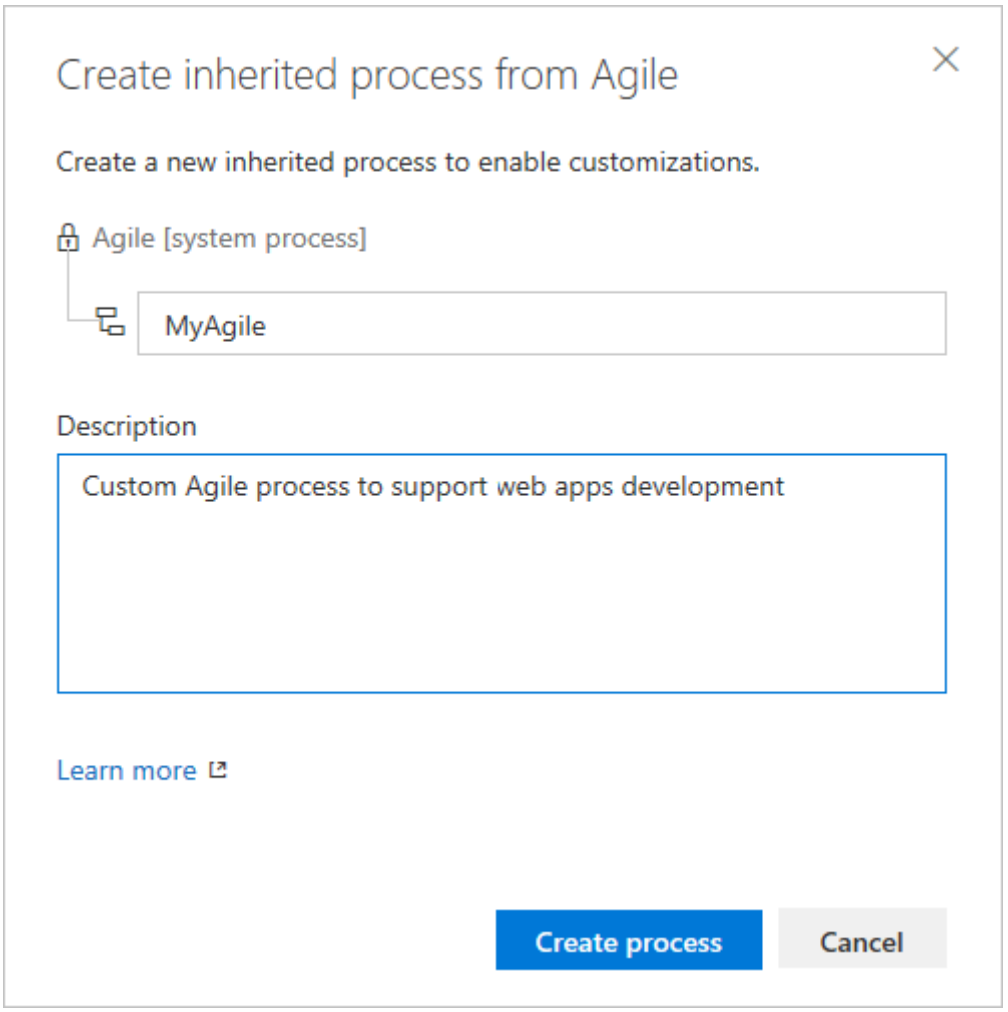
3. Select **Process** > **...** (More actions) > **Create inherited process**. Choose the same system process—Agile, Basic, Scrum, or CMMI—that was used to create the project that you want to customize.

   In the following example, we create an inherited process from the Agile system process.



If you don't have access to these options, ask a member of your **Project Collection Administrators** group to grant you permissions. To find a member, see Look up a Project Collection Administrator.

4. Enter a name for your process and optionally a description. Process names must be unique and no more than 128 characters. For other restrictions, see Create and manage inheritance processes, Process name restrictions.



After you define the inherited process, you can do the following tasks:

- Customize a project using an inherited process
- Create a project that uses the inherited process
- Change project to use the inherited process

Change a project's process

You can change a project's process from one inherited process to another with the following methods:

- **Switch within the same base process:** Move a project between processes that share the same base, such as Agile or Scrum.
- **Migrate to a different process model:** Change the project's process model, for instance, from Agile to Scrum or Basic to Agile.

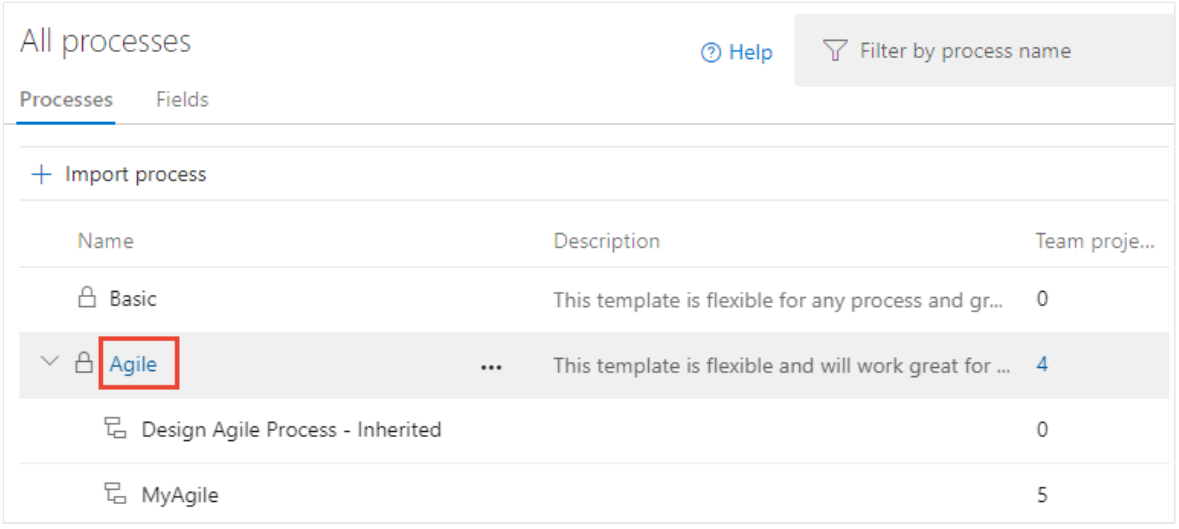
We provide detailed steps for the second method, covering the following common scenarios of process change:

- Scrum to Agile
- Agile to Scrum

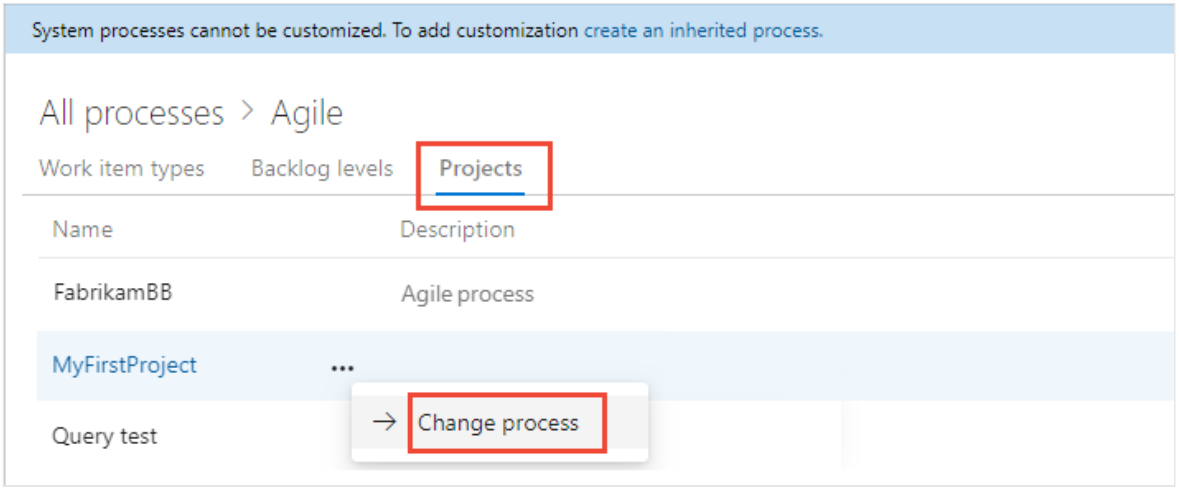
- Basic to Agile

> **Note**
>
> - You can change the process of a project as long as you don't have any undeleted work items of a custom work item type that isn't also defined in the target process.
> - If you change a project to a system process or other inherited process that doesn't contain the same custom fields, data is still maintained. But, the custom fields that aren't represented in the current process won't appear on the work item form. You can still access the field data through a query or REST APIs. These fields are essentially locked from changes and appear as read-only values.

1. Select your project's process. For example, to change a project from Agile to Scrum, then choose the **Agile** process.



2. Select **Projects** > the ··· actions icon for the project > **Change process**.

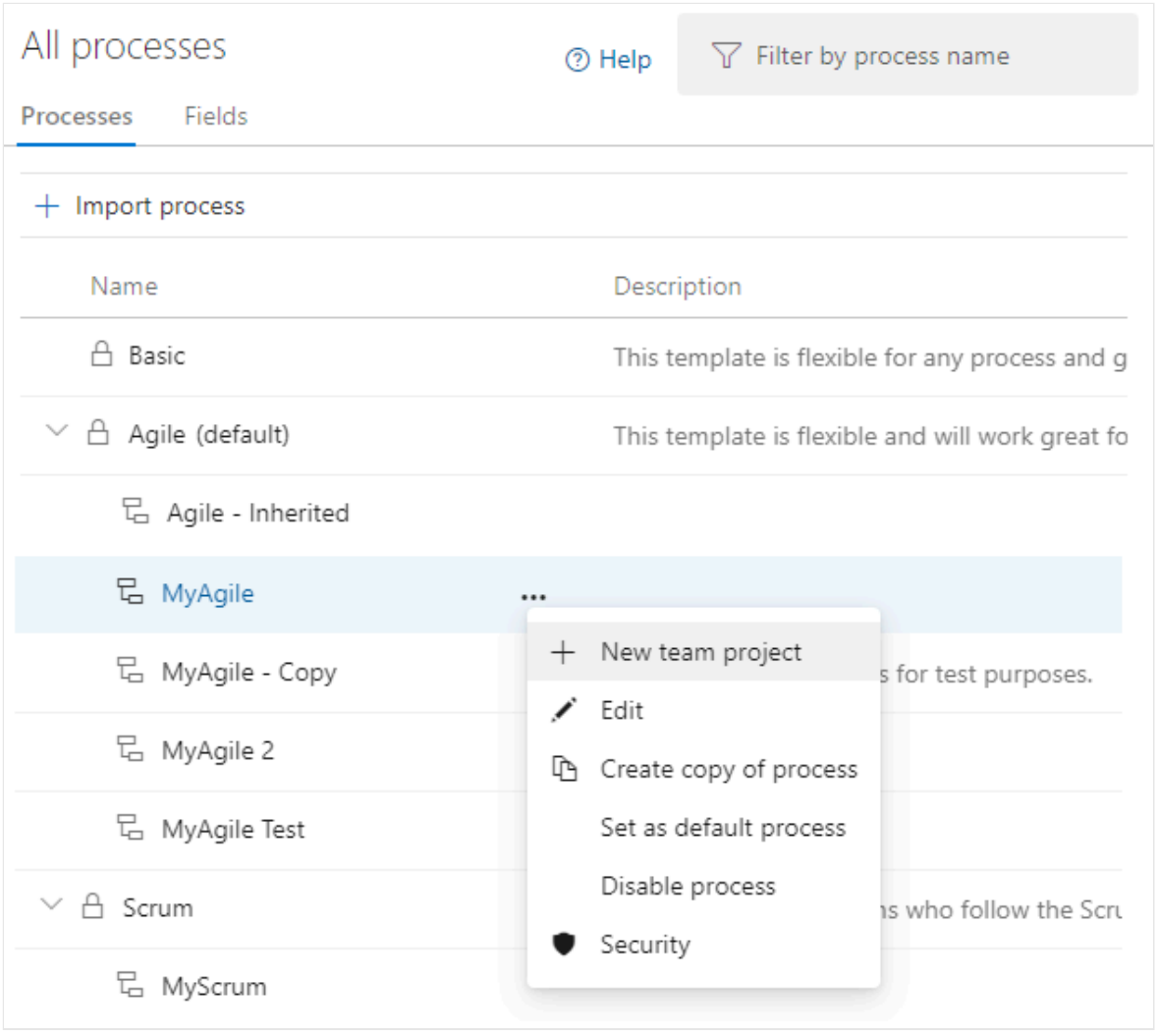


3. Complete the steps in the wizard.

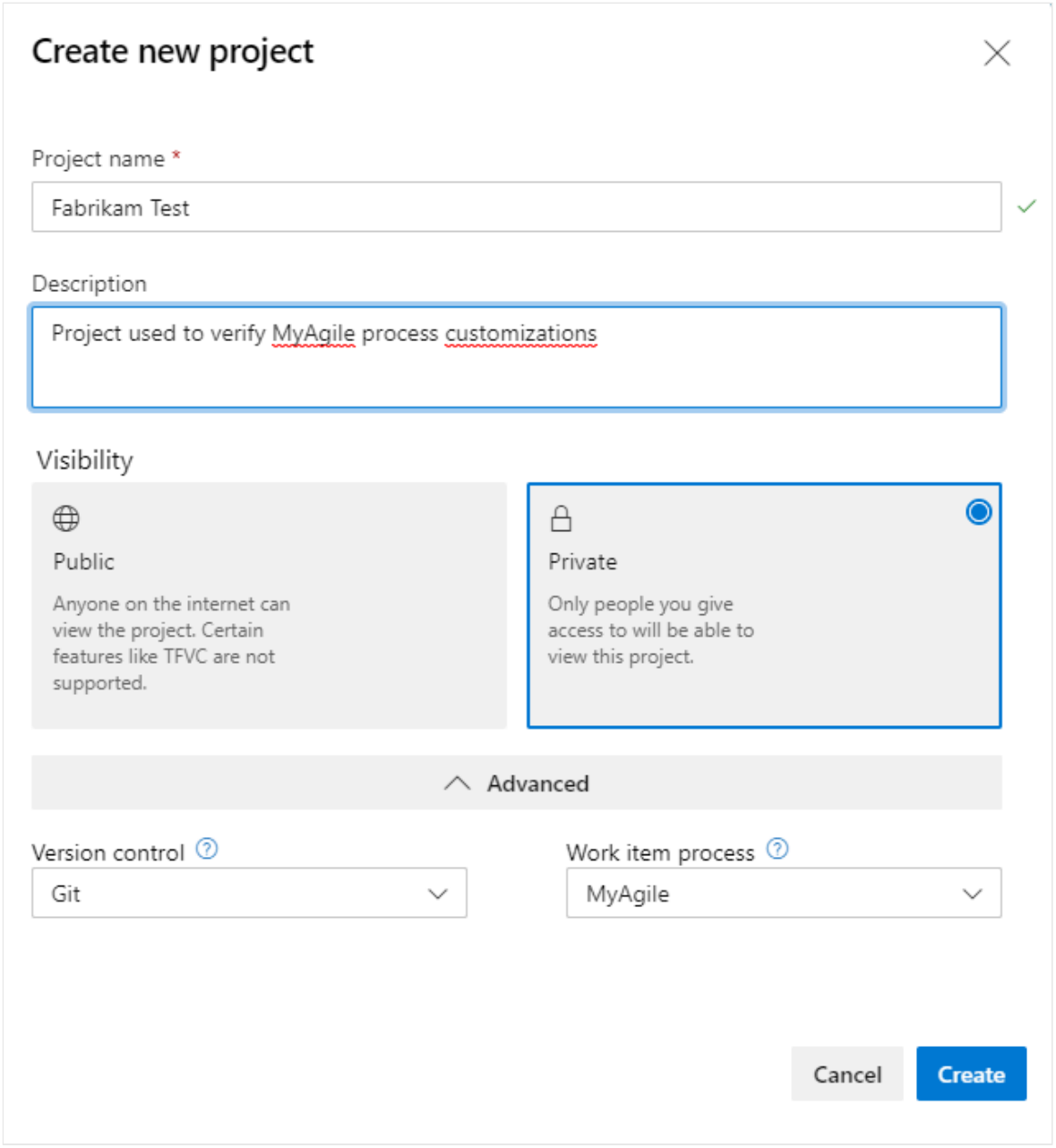
> **Important**
>
> When you switch a project to an inherited process, some Agile tools or work items might become invalid. For example:
>
> - If you designate a field as required, work items lacking that field display an error message. Resolve these errors to proceed with further changes and save the work item.
> - When you add or modify workflow states for a WIT visible on your board, remember to update the board column configurations for all teams within the project.

Create a project from a process

1. Open the ... context menu for the process you want to use and select **New team project**.



2. Enter your project information, and then select **Create**. For more information, see Create a project.

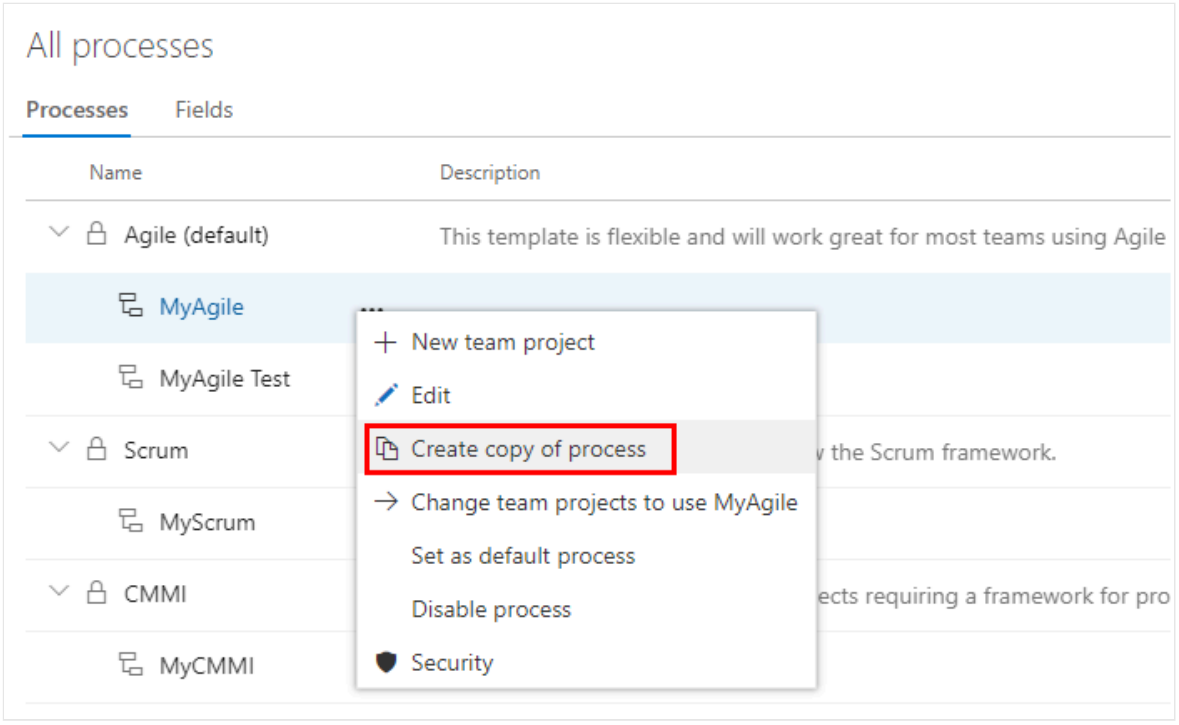


Copy a process

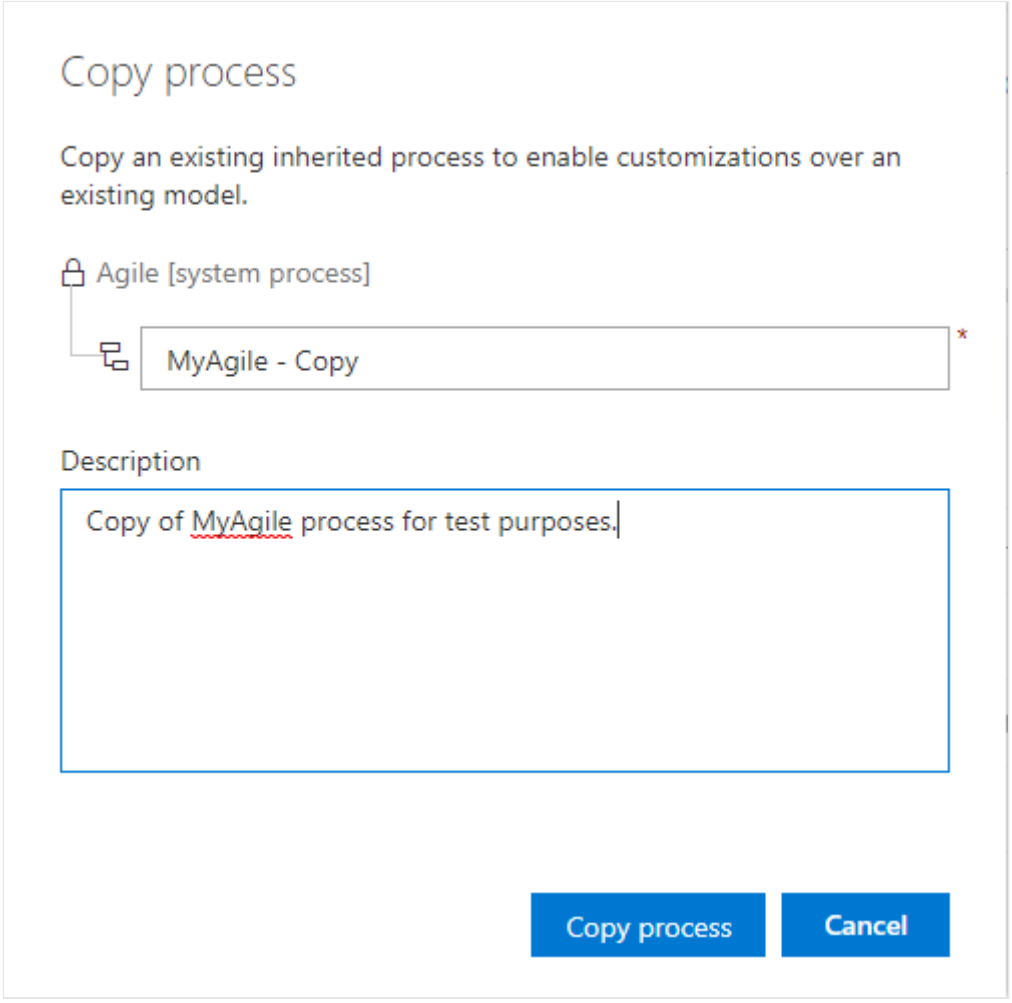
Before you implement customizations across your organization, it's essential to test them by doing the following steps.

> **Tip**
>
> If you modify a process used by multiple projects, each project immediately reflects the incremental process change. To bundle process changes before rolling them out to all projects, do the following steps.

1. From the Process page, open the ... context menu for the process and select **Create copy of process**.



2. Enter a name and optional description for the copied process and select **Copy process**.



3. Make your changes to the copied process. Since no project is using this process, these changes don't affect any projects.

4. Verify your changes by creating a test project based on the copied and updated process. If you already created a test project, select **Change project to use *ProcessName***.

5. Roll out your updates by changing the process of the projects that need the new changes. Select **Change project to use *ProcessName***.

6. Disable or delete the original process.

Enable/disable a process

To prevent projects being created from a specific process, you can disable it. You might choose this option when you want to apply several customizations and don't want the process used until they're complete. Or, you might retire use of a specific process in favor of moving projects to a new process.

All system processes and newly created inherited processes are enabled by default. To disable or enable a process, open the ... context menu for the process and choose **Disable process** or **Enable process**.

Set the default process

To have an inherited process preselected for other projects you plan to create, set it as the default. This action ensures that any new projects automatically use the inherited process you choose.

To set a process as the default, open the ... context menu for the inherited process and choose **Set as default process**. This option isn't available with any of the system processes.

Project Collection Administrators can add projects from the **Projects** page.

Next steps

Add and manage fields for an inherited process or

Add and manage work item types

Related articles

- About process customization and inherited processes
- Customize a project using an inherited process.
- Use the REST API for processes.

Feedback

Was this page helpful? Yes No

Provide product feedback

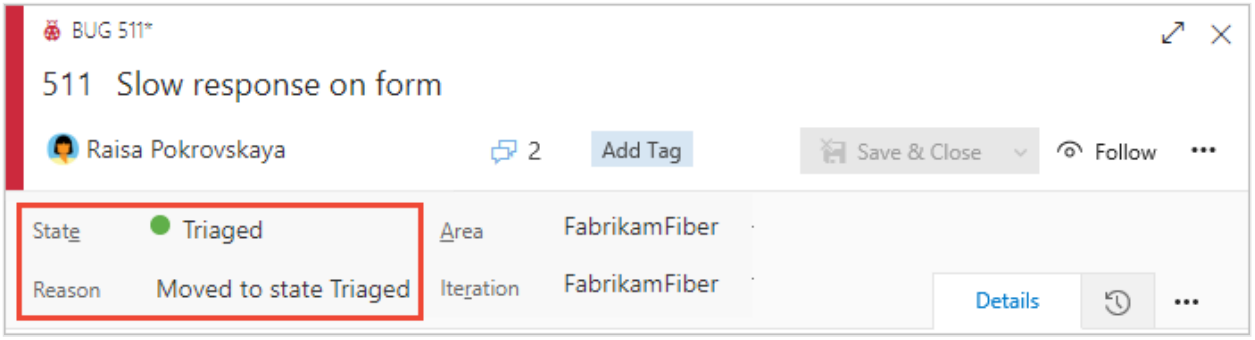
Customize the workflow (Inheritance process)

Article • 10/21/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Each work item type (WIT) has an associated workflow that tracks the status of work from creation to completion. To align with your business and team processes, you can add custom states to most work item types. For example, you might add a *Triaged* state for bugs or a *Design* state for features or user stories.

In this article, you customize the Bug WIT to include a Triaged state. The state and reason fields are displayed in the header area of the work item form.



For documentation on the workflow for build and release DevOps tasks, see YAML vs Classic Pipelines.

Supported customizations

You can customize the workflow of any work item type (WIT) by hiding inherited states or adding custom states. Inherited states vary based on the system process that you selected to create your custom process. The options are *Agile*, *Basic*, *Scrum*, or *Capability Maturity Model Integration (CMMI)*. For more information, see Workflow states, transitions, and reasons.

Each default workflow for each WIT defines between two and four states and specifies the following workflow operations:

- Forward and backward transitions between each state. For example, the Basic process Issue WIT includes three states—**To Do**, **Doing**, and **Done**.
- Default reasons for each state transition

State types

Supported customizations

 Inherited states

- Hide or unhide a state
- Add rules when changing a workflow state

Custom states

- Add a workflow state
- Edit a workflow state (change color or category)
- Remove a workflow state
- Add rules when changing a workflow state

Workflow states must conform to the following rules

- Define at least one state for either the *Proposed* or *In Progress* State categories.

  > **Note**
  >
  > Before you add a workflow state, see About workflow states in backlogs and boards to learn how workflow states map to state categories.

- Define at least two workflow States.
- Define a maximum of 32 workflow States per work item type.

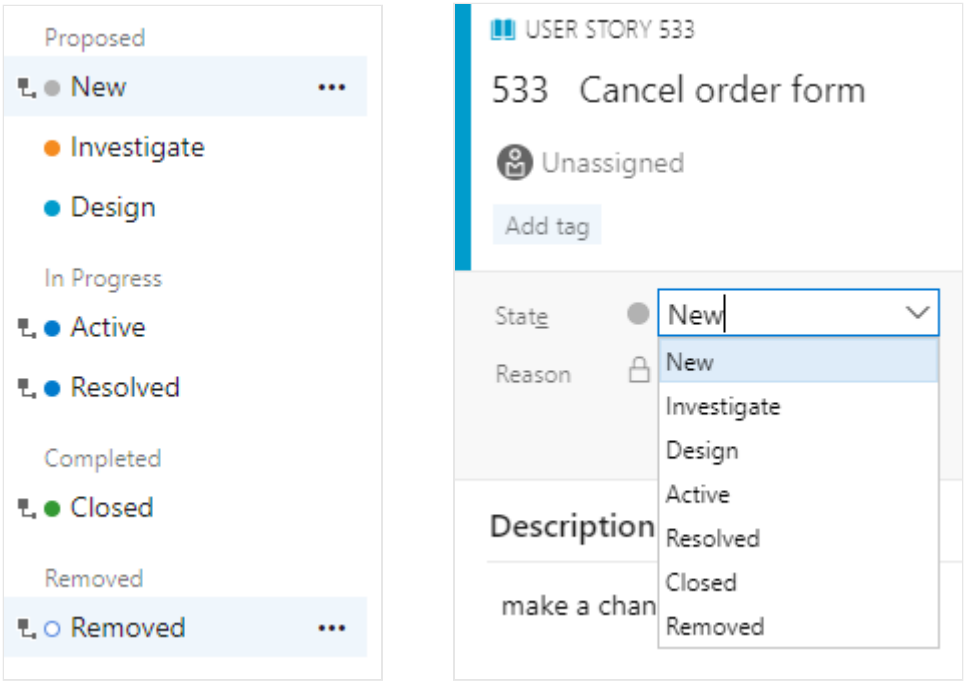
Unsupported workflow customizations

- Hide inherited states if you don't want them visible (you can't change their name, color, or category).
- Ensure only one state exists in the *Completed* state category. Adding a custom state to this category removes or hides any other state.
- Keep the name of custom states as is; you can't change them.
- Use default reasons for state transitions, such as *Moved to state Triaged* and *Moved out of state Triaged*; you can't specify custom reasons.
- Accept the default location of the State and Reason fields on the form; you can't change their placement.
- Use the default state category names; you can't customize them.

State dropdown menu sequence

The **State** dropdown menu lists states in the order you define within each state category. For newly added work items, the first state in the *Proposed* category is assigned as the default state.

The following image illustrates the state sequence defined for a User Story and its corresponding dropdown menu.



Within each category, you can move custom states up or down.

Affect to teams with workflow changes

Update board configuration

Teams should update their board configuration when making the following customizations:

- Add a custom state.
- Change the category of a custom state.
- Add a custom or inherited work item type to a backlog level. See Customize backlogs and boards.

Taskboard configuration

Teams should update their board configuration when making the following customizations:

- Add states to the task WIT, which adds columns to the Taskboard.
- Track bugs along with tasks, adding states to the bug WIT, which also adds columns to the Taskboard.
- Add the same states to both task and bug work item types, which updates the status consistently and minimize the number of columns added.

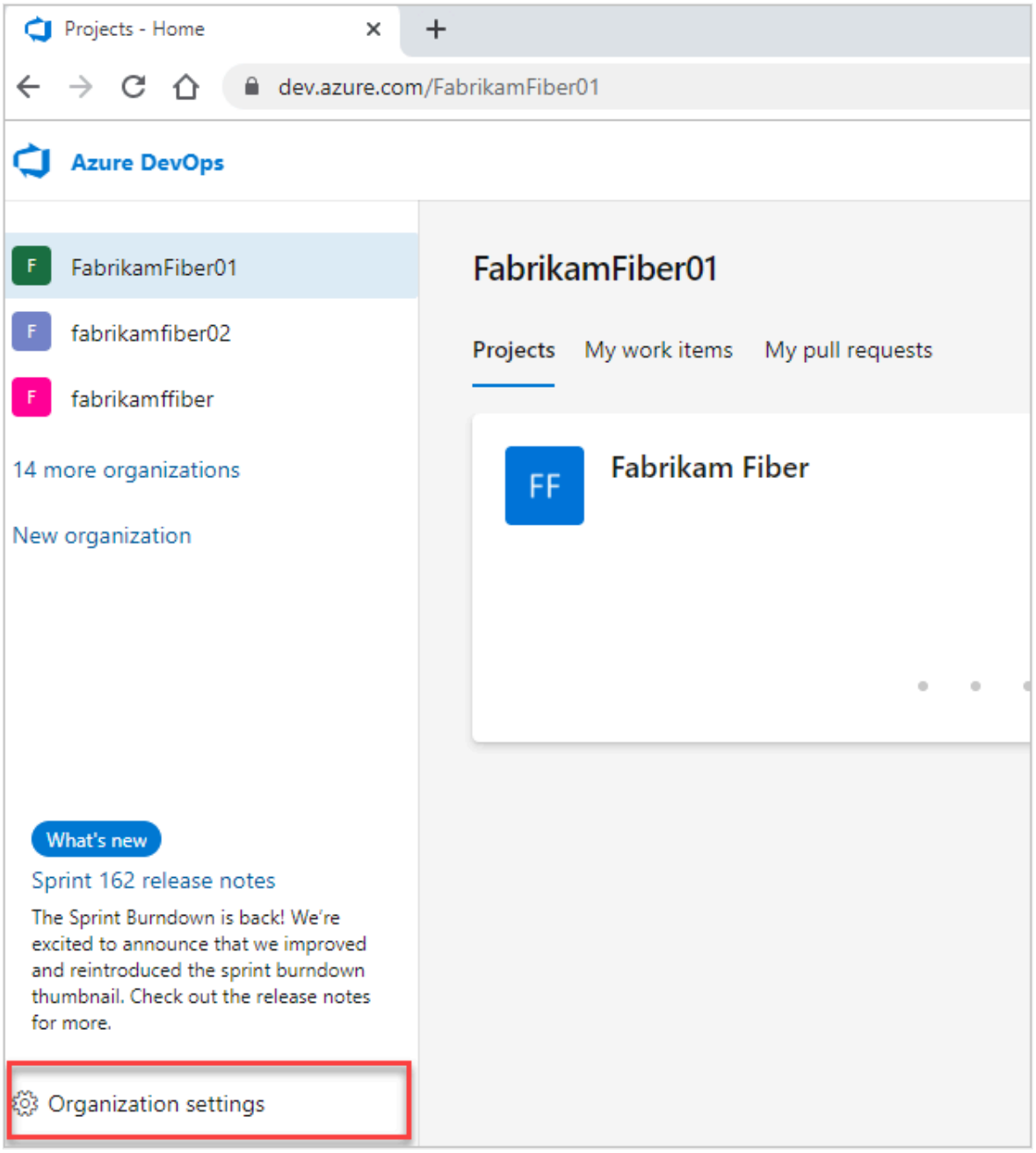
Prerequisites

For guidance on tailoring Azure Boards to align with your specific business requirements, see About configuring and customizing Azure Boards.

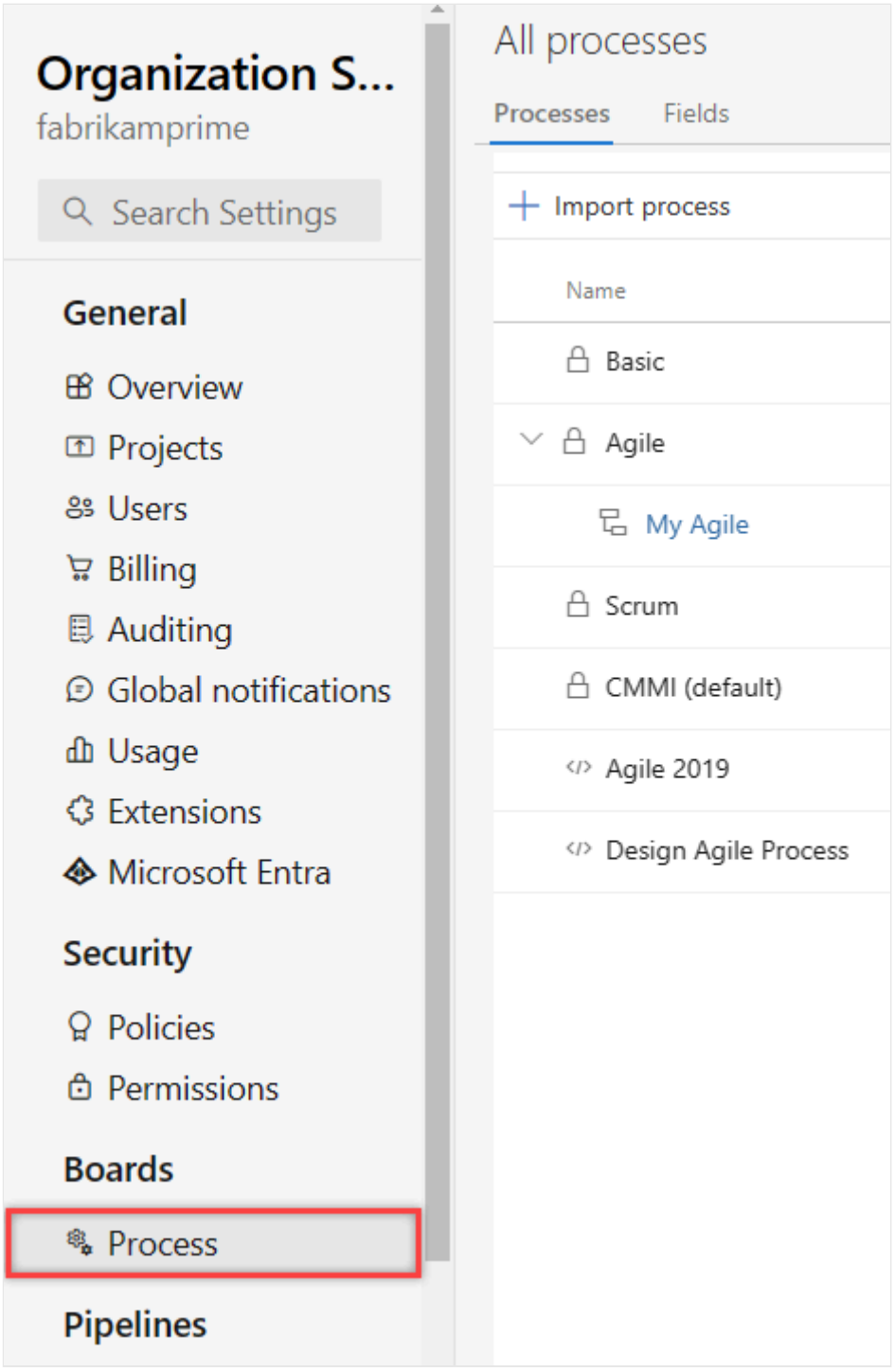
- **Organization requirement**: Ensure you have an organization in Azure DevOps.
- **Permissions**:
  - Be a member of the **Project Collection Administrators** group.
  - Have collection-level permissions such as **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**.
  - These permissions allow you to modify processes and fields within your organization.

Open organization process settings

1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`).
2. Select **Organization settings**.



3. Select **Process**.

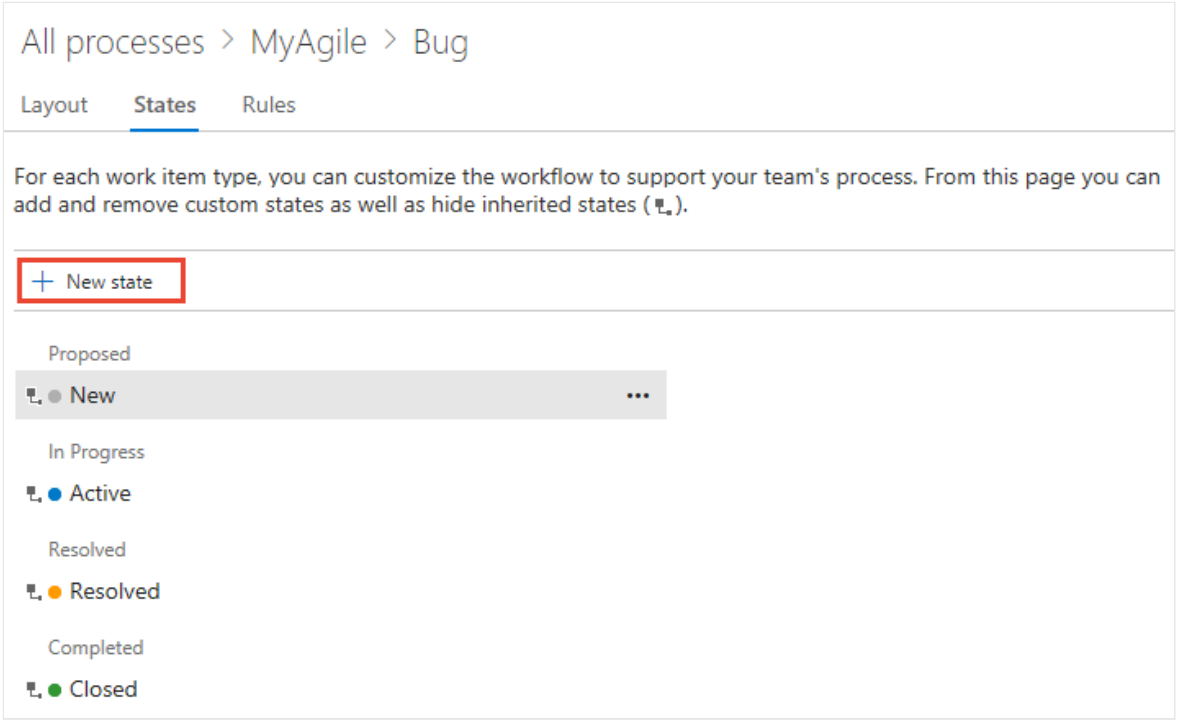


> **Note**
>
> When you customize an inherited process, any projects using that process automatically reflect the customizations. To ensure a smooth transition, we recommend creating a test process and project, which allows you to test your customizations before you implement them organization-wide. For more information, see Create and manage inherited processes.

Add a workflow state

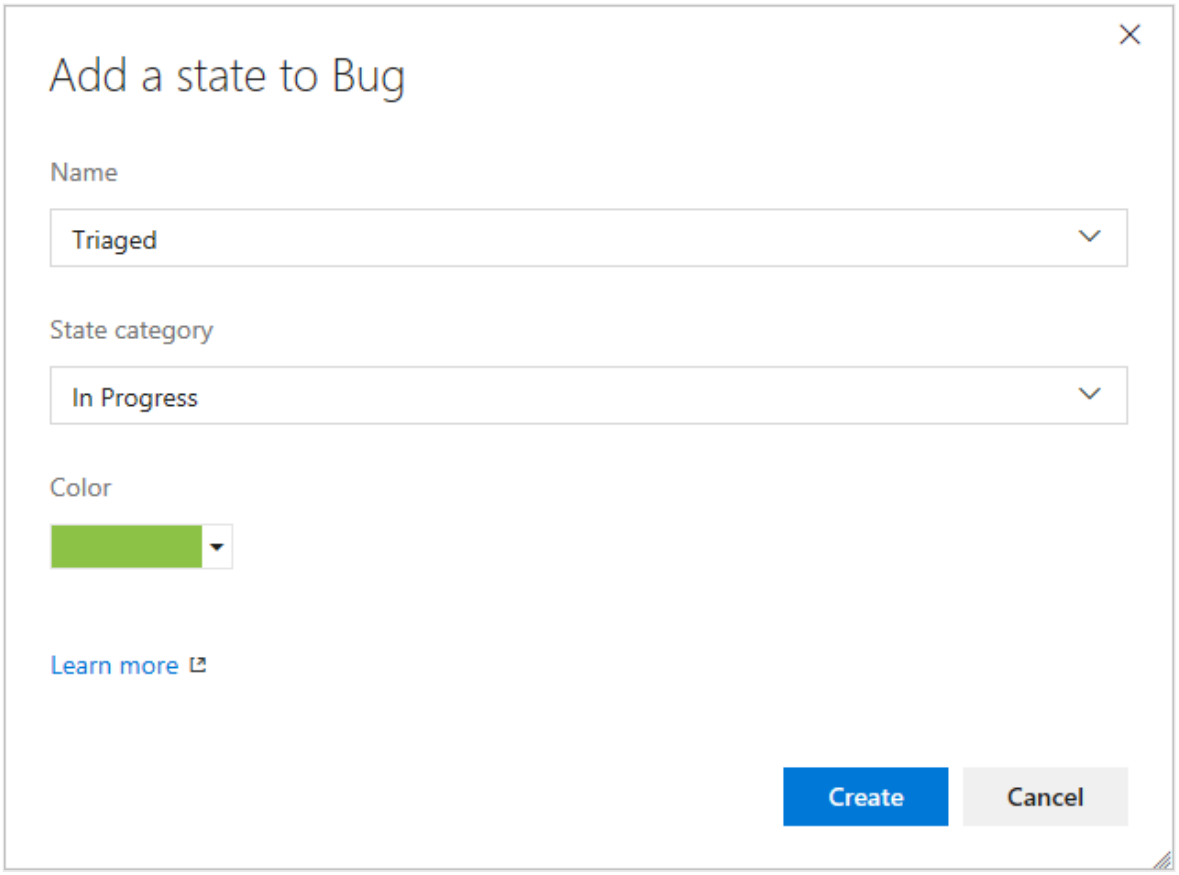
States you add appear in the dropdown menu for the States field shown in work item forms and the query editor. A transition to and from the State you add is created to every other State. Default reasons are defined, such as *Moved to state Triaged* and *Moved out of state Triaged*.

1. From the **Work Item Types** page, choose the work item type you want to modify, choose **States**, and then choose **New State**.



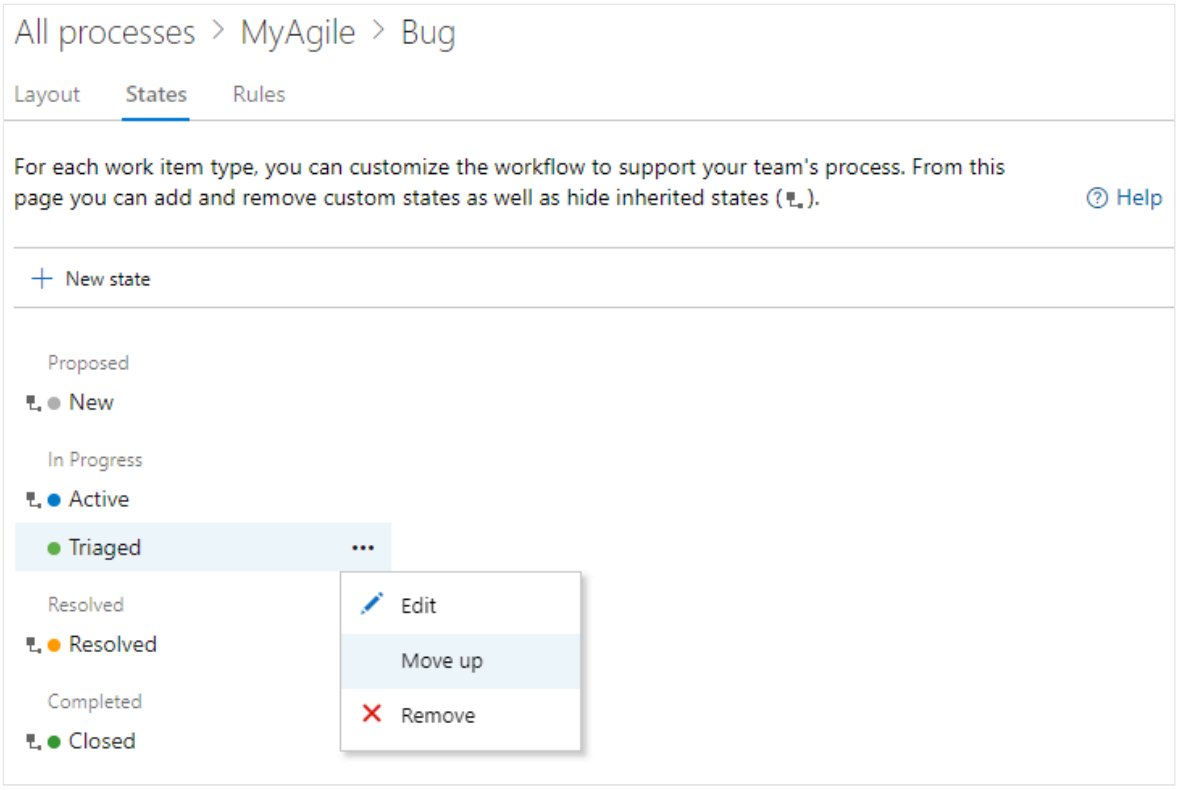
   If the **New state** option is disabled, you don't have the necessary permissions to edit the process. See Customize an inherited process.

2. Enter the name of the State, choose its category and color, and then select **Save**. The color you specify appears throughout the product including on the work item form and when the State field appears on a backlog, boards, query results, and more.



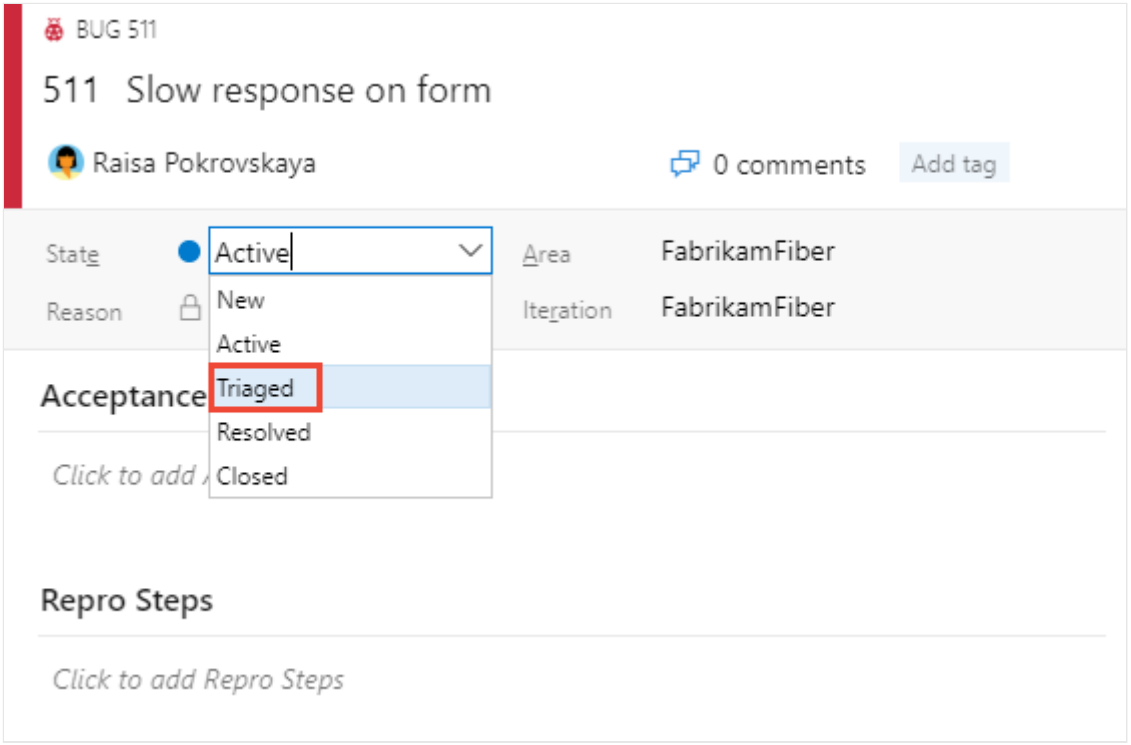
> **Note**
>
> Any workflow state you add to the *In Progress* or *Resolved* state categories will cause the **Activated By/Activated Date** and **Resolved By/Resolved Date** fields to update with workflow state changes in and out of these categories. For more information, see Activated By/Date and Resolved By/Date fields.

3. (Optional) To change the sequence of the State within the dropdown menu, choose the ··· context menu icon and choose **Move up** or **Move down**.



4. When you're done adding states for the WIT, verify your changes by refreshing your browser and open a work item of the type you customized.

   Here is the State dropdown menu with Triaged selected.

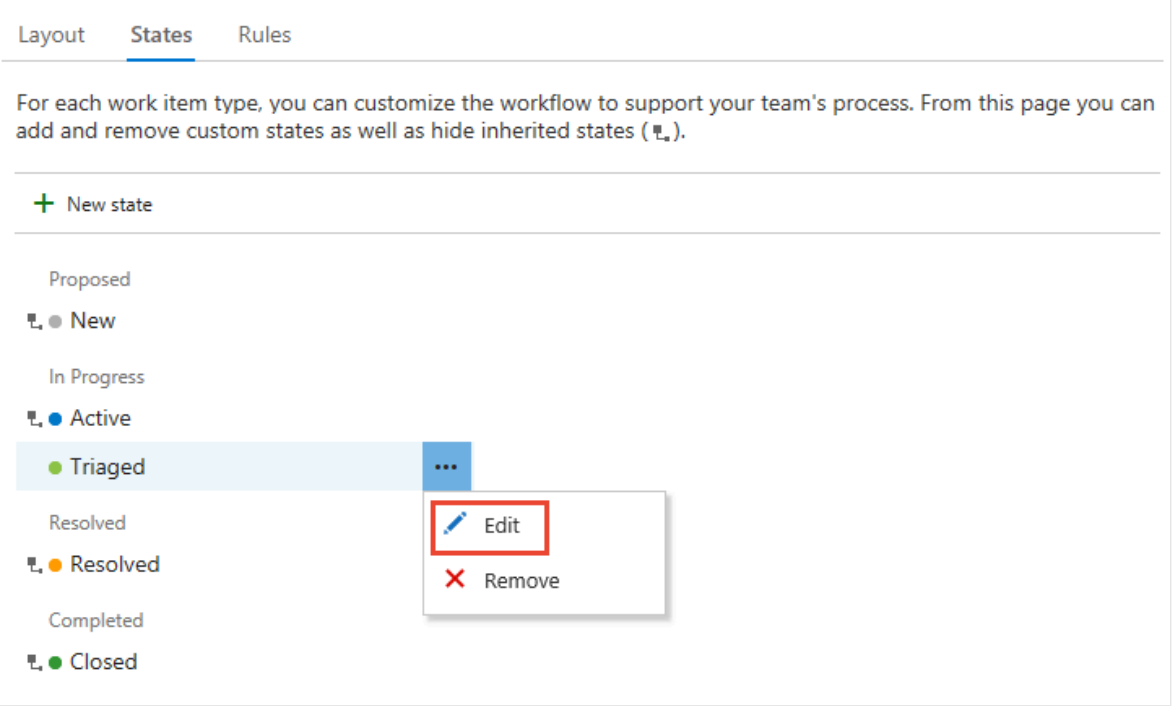


5. When you add a State to a WIT, which is associated with a backlog level, each team that uses the board needs to update their column settings. See Manage columns on your board.

Edit a state

You can edit the category or the color of a custom state. However, you can't change the name of the custom state.

1. Select **Edit** from the ... context menu for the state you want to modify.



2. Modify the category or color, and then choose **Save**.

3. If you change the category, teams that use the board need to update their column settings. See Manage columns on your board.

Hide or remove a custom state

When you hide or remove a state:

- The state no longer appears in the State dropdown menu for the WIT
- No changes occur to the work item history
- Existing work items maintain their state value, but are in an invalid state. If you want to make a change to the work item, you must first update the state value.

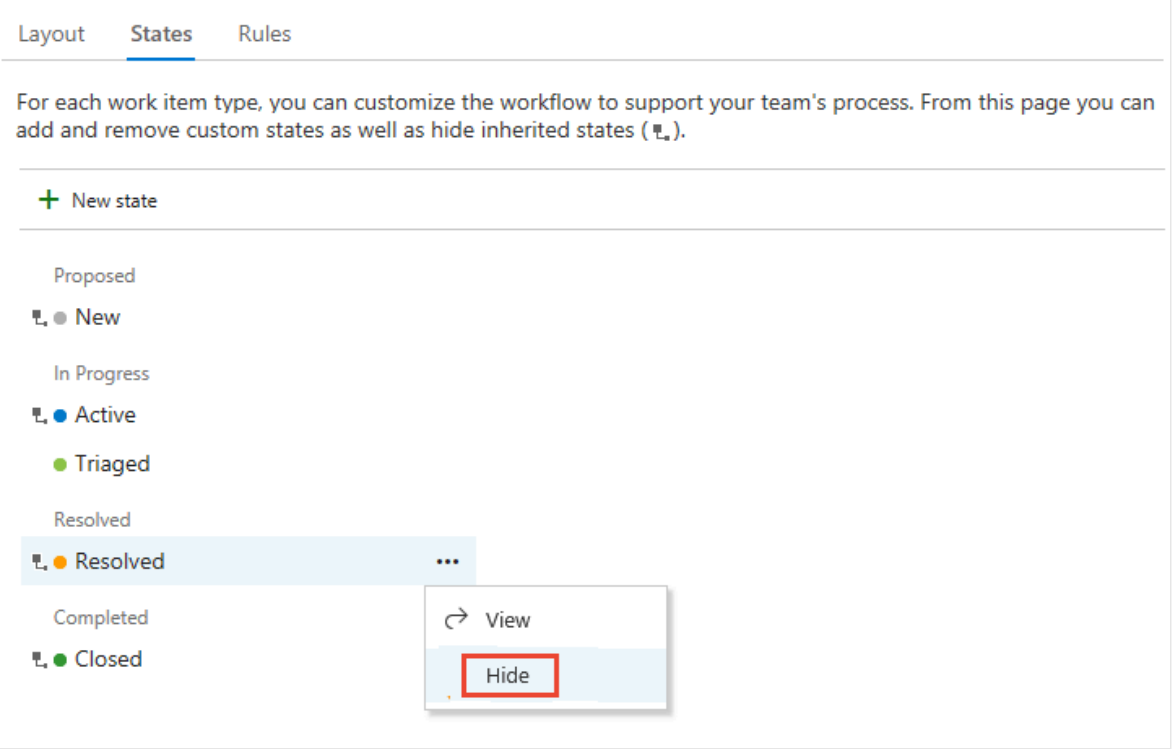
  You might want to create a query and do a bulk update to move the affected work items into a valid state. If you add the state back to the work item type, the work items revert to a valid state.

Hide or unhide an inherited state

You can hide an inherited state that your team doesn't use in its workflow process. However, you must have at least one state defined for each category.

1. Open the ... context menu for the state you want to hide and choose the **Hide** option.

   This example hides the Resolved state for the Bug WIT.



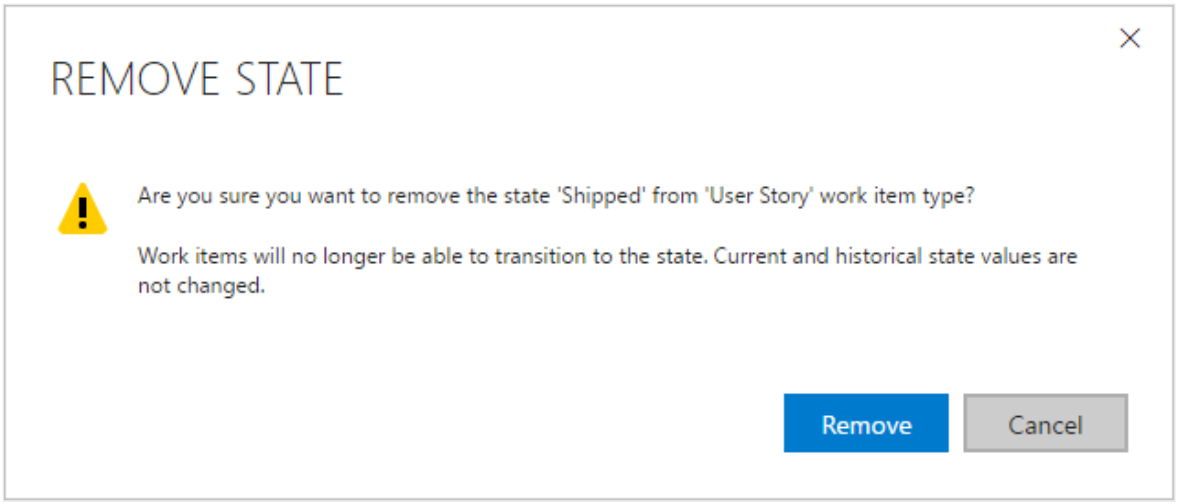
> **Note**
>
> If you hide the state of a WIT tracked on a board, each team that uses the board needs to update their column settings. See Manage columns on your board.

2. To unhide, open the ... context menu and choose the **Unhide** option.

Remove a custom state

1. Open the ... context menu for the state you want to remove, and choose **Remove**. You can only remove a custom state.

2. From the **Remove State** dialog, select **Remove**.



View the State workflow model

You can view the State workflow model by installing the State Model Visualization Marketplace extension. This extension adds a new hub under Boards labeled **State Visualizer**. On that page you can choose a work item type and view the workflow state model.

> **Note**
>
> The State Model Visualization extension is not supported by Azure Boards or the product team. For questions, suggestions, or issues, please visit the extension page.

For example, you can customize the Bug workflow to have a **Triaged** state and all states can transition from one state to another.

You can zoom in and zoom out of the view. Also, you can move the state nodes around to gain a better view of the state model.

Next steps

Review changes made to an inherited process through the audit log

Related articles

- Learn about workflow states in backlogs and boards
- Add and manage work item types
- Add and manage fields

- Customize the web layout for a work item type
- Customize a project using an inherited process
- Manage columns on your board

Feedback

Was this page helpful? Yes No

Provide product feedback

Apply rules to workflow states (Inheritance process)

Article • 09/11/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

After you add or modify your workflow states for a work item type, define rules that apply based on the workflow state change. Adding rules to workflow states supports the following scenarios:

- Support an approval process
- Prevent unauthorized users from setting an invalid state
- Make a field required or read-only or another value based on State changes
- Restrict transition from one state to another
- Restrict or allow State transitions to specific users or groups
- Maintain a controlled workflow process, supporting auditing requirements
- Automate closure of parent work items

Prerequisites

To apply rules to workflow states in Azure DevOps, you need specific permissions and access levels:

- **Permissions:**
  - Be a **Project Administrator** to manage security groups and permissions at the project level, which includes setting rules for workflow states.
  - Have **Work Item Tracking permission**, which allows you to manage the work tracking area, which can be granted to members of the Project Administrators group or through specific permissions.

- **Access levels:**
  - Have **Basic** access, which is typically sufficient for most users who need to manage work items and apply rules to workflow states.

Understand workflow rules

The following table outlines the three groups of workflow rules you can define:

1. **Standard actions:**

- Apply when a work item is created, in a selected state, or moved from one state to another.
- Actions include setting the value of a field, making a field read-only, or making a field required.
- You can specify one or two conditions and several actions.

2. **Restricting state transitions (group 1):**

   - Specify one condition indicating the state a work item moved from.
   - Define actions to restrict transitions from that state to other states.

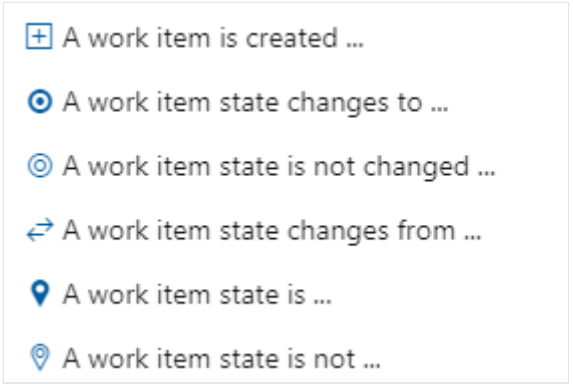
3. **Restricting state transitions (group 2):**

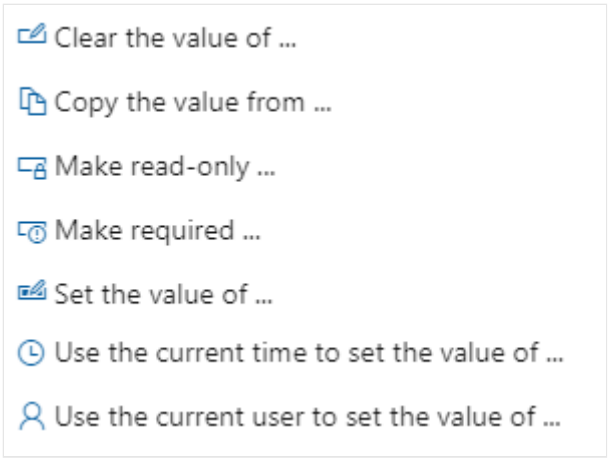
   - Similar to the first group, specify one condition indicating the state a work item moved from.
   - Define actions to restrict transitions from that state to other states.

Condition

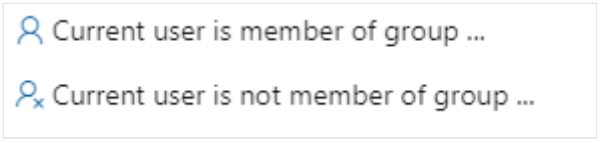
Supported Actions

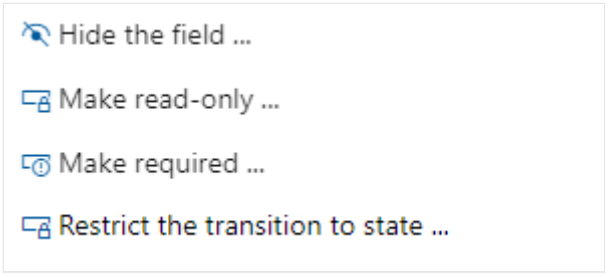
Set field value or make read-only/required based on State





Based on and user or group membership, set a field attribute or restrict a State transition





> **Note**
>
> When you customize an inherited process, any projects using that process automatically reflect the customizations. To ensure a smooth transition, we recommend creating a test process and project, which allows you to test your customizations before you implement them organization-wide. For more information, see Create and manage inherited processes.

Understand workflow state and rule limits

Workflow rules get applied when you add or modify work items through any of the following interfaces:

- **Web portal:** Work item form, bulk updates, updates in query view
- **Web portal:** Board or Taskboard, move work item to column
- **Visual Studio 2017** and earlier versions, work item form
- **CSV file format:** Bulk import or update
- **Excel:** Bulk import or update
- **REST API:** Add or modify work items

The following table summarizes the workflow state and rule limits for the Inheritance process.

Expand table

Object	Inheritance limit
Work item types defined for a process	64
Workflow states defined for a work item type	32
Rules defined for a work item type	1024

When you define workflow states and rules, follow these guidelines to minimize performance issues:

- **Limit the number of rules for a WIT:** While you can create multiple rules for a work item type (WIT), more rules can negatively affect performance when users add or modify work items. The system validates all rules associated with the fields for the work item type when users save work items. In some cases, the rule validation expression might become too complex for SQL to evaluate.
- **Limit the number of custom work item types:** Reducing the number of custom work item types can help maintain optimal performance.

Define a rule

Before you define a rule based on workflow states, ensure the following elements are in place:

- **Workflow states:** Define the workflow states as described in Customize a workflow.
- **Custom fields:** If your rule requires a custom field, add it to the work item type as described in Add and manage fields.
- **Security groups:** If your rule requires a security group to grant or restrict changes that are based on user or group membership, define the security group as described in Add or remove users or groups, manage security groups.

For more information about defining rules, see Add a custom rule.

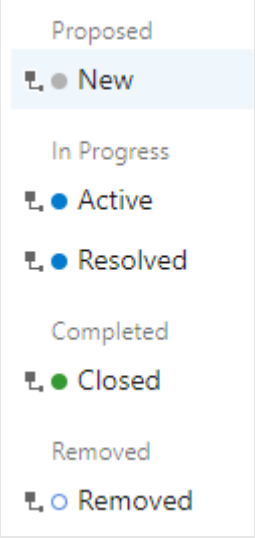
Set field value or make field read-only or required

With the first grouping of rules, you can specify one or two conditions and up to 10 actions per rule.

Example of ensuring team lead approval before active work

In this example, development teams want to ensure that no User Story gets worked on until approved by a team lead. The default workflow states are used, with the addition of a custom field, *Approved By*, and a security group, *Team Leads Group*.

Default workflow states



Rule requirements

To ensure approval before active work, define the following rules:

- Require the *Approved By* field to be filled in when the State moves from *New* to *Active*
- Restrict users who aren't in the *Team Leads Group* from filling in the *Approved By* field
- Clear the *Approved By* field when the State moves to *New* or *Removed*

Rule definitions

The rule requirements translate to the following four rule definitions.

Rule name

Condition

Actions

Approved By cleared when New

When `A work item state changes to New`

Then `Clear the value of Approved By`

Approved By cleared when Removed

When `A work item state changes to Removed`

Then `Clear the value of Approved By`

Approved By Read-only

When `Current user is not member of group Team Leads Group`

Then `Make read-only Approved By`

Approved By required

When `A work item state changes from New to Active`

Then `Make required Approved By`

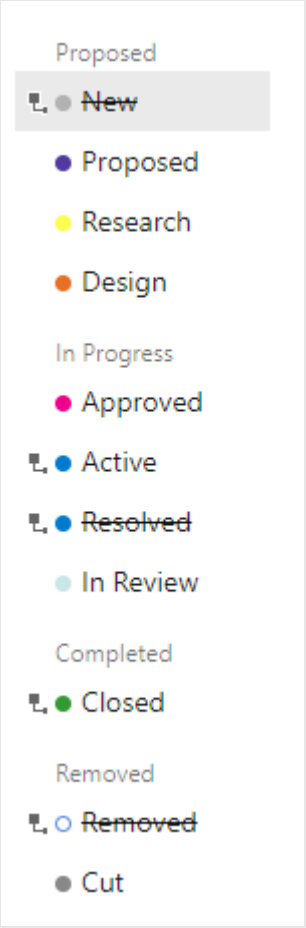
Restrict state transitions

When you specify the condition, `A work item state moved from ...`, you can specify only that condition. You can specify up to 10 actions.

Example of restricting state transitions and Approved state

The following workflow states are defined for the User Story. The *New*, *Resolved*, and *Removed* inherited states are hidden. Instead, *Proposed*, *In Review*, and *Cut* States are

used. In addition, three more States are defined: *Investigate*, *Design*, and *Approved*. These States should follow the sequence as shown in the following image.



Without any restrictions, users can move from one State to any other State, both forward and backward within the sequence.

Rule requirements

To support a more controlled workflow, the business group decided to institute rules that support the following forward and reverse state transitions on the User Story work item type.

Expand table

State	Transition rule
Proposed	Can only move to *Research* and *Cut*
Research	Can only move to *Design* and *Cut*
Design	Can only move to *Research*, *Approved*, and *Cut*
Approved	Can only move to *Design*, *Active*, and *Cut*

State	Transition rule
Active	Can only move to *In Review*
In Review	Can only move to *Active* (More work found), *Closed* or *Cut*
Closed	Can move to *Research*, *Design*, *Active*, *In Review* (Allows for cases where user closed the work item in error)
Cut	can only move to *Proposed*

> **Note**
>
> When you restrict state transitions, account for cases where a user might move a state in error. Ensure users can recover gracefully.

Additionally, the business group wants to apply the following rules for required fields:

- Require the *Approved By* field to be filled in when the state moves from *Approved* to *Active*.
- Allow only users in the *Authorized Approvers* group to fill in the *Approved By* field.
- Clear the *Approved By* field when the state moves to *Cut*.
- Require the *Acceptance Criteria* field to be filled in when the state moves to *Active*.

Rule definitions

To implement the previously mentioned restrictions, the process administrator adds a custom *Approved By* identity field, an *Authorized Approvers* security group, and the following rules.

Rule name

Condition

Actions

Proposed state

When `A work item state moved from Proposed`

Then `Restrict the state transition to Design`
And `Restrict the state transition to Approved`
And `Restrict the state transition to Active`
And `Restrict the state transition to In Review`
And `Restrict the state transition to Closed`

Research state

When `A work item state moved from Research`

Then `Restrict the state transition to Proposed`
And `Restrict the state transition to Approved`
And `Restrict the state transition to Active`
And `Restrict the state transition to In Review`
And `Restrict the state transition to Closed`

Design state

When `A work item state moved from Design`

Then `Restrict the state transition to Proposed`
And `Restrict the state transition to Research`
And `Restrict the state transition to Active`
And `Restrict the state transition to In Review`
And `Restrict the state transition to Closed`

Approved state

When `A work item state moved from Approved`

Then `Restrict the state transition to Proposed`
And `Restrict the state transition to Research`
And `Restrict the state transition to Design`
And `Restrict the state transition to In Review`
And `Restrict the state transition to Closed`

Active state

When `A work item state moved from Active`

Then `Restrict the state transition to Proposed`
And `Restrict the state transition to Research`
And `Restrict the state transition to Design`
And `Restrict the state transition to Approved`
And `Restrict the state transition to Closed`

In Review state

When `A work item state moved from In Review`

Then `Restrict the state transition to Proposed`
And `Restrict the state transition to Research`
And `Restrict the state transition to Design`
And `Restrict the state transition to Approved`

Closed state

When `A work item state moved from Closed`

Then `Restrict the state transition to Proposed`
And `Restrict the state transition to Cut`

Cut state

When `A work item state moved from Cut`

Then `Restrict the state transition to Research`
And `Restrict the state transition to Design`
And `Restrict the state transition to Approved`
And `Restrict the state transition to Active`
And `Restrict the state transition to In Review`
And `Restrict the state transition to Closed`

Approved state required fields

When `A work item changes from Approved to Active`

Then `Make required Acceptance Criteria`
And `Make required Approved By`

Authorized Approvers

When `Current user is not a member of Authorized Approvers`

Then `Make read-only Approved By`

Clear Approved By field

When `A work item state changes to Cut`

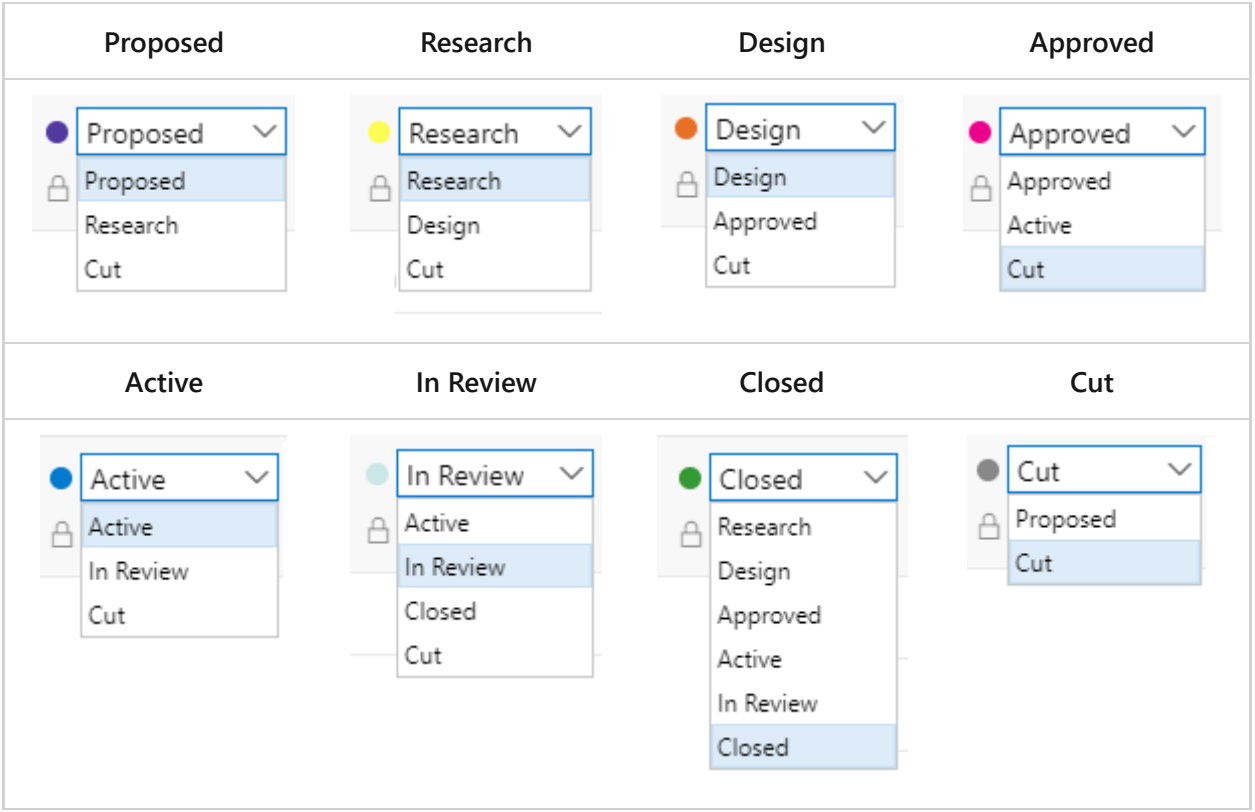
Then `Clear the value of Approved By`

Verify state transition restrictions

Once you define the rules for the process and update the project, refresh your browser. Verify the operations through the work item form and the browser.

For the rules defined in the previous table, check the State drop-down menus. Open the board and ensure you can move from one state to another.

Expand table



Restrict state transition based on user or group membership

When you specify one of the two conditions based on user or group membership, `Current user is member of group ...` or `Current user is not member of group ...`, you can only specify one condition. Additionally, if you specify the action `Restrict the transition to state...`, you can only specify one action.

> **Note**
>
> Work items are subject to rules applied to them. Conditional rules based on user or group membership are cached for your web browser. If you find yourself restricted to update a work item, you may have encountered one of these rules. If you believe you've encountered an issue that doesn't apply to you, see Work item form IndexDB caching issues.

Automate state transitions of parent work items

To automate State transitions for parent work items that are based on the State assignments of their child work items, see Automate work item state transitions.

Automate reassignment based on state change

The Agile process bug work item type previously had a rule that reassigned the bug to its creator. We removed this rule from the default system process. You can reinstate the rule or add a similar rule to other work item types using the following condition and action:

When `A work item state changes to` *Resolved* **Then** `Copy the value from` *Created By* **to** *Assigned To*.

Related articles

- Add a custom rule
- Work item form caching
- Access, export, and filter audit logs
- Customize a workflow

Feedback

Was this page helpful? Yes No

Provide product feedback

Add a rule to a work item type (Inheritance process)

Article • 02/24/2023

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Custom rules provide support for a number of business use cases, allowing you to go beyond setting a default value for a field or make it required. Rules allow you to clear the value of a field, copy a value into a field, and apply values based on dependencies between different fields' values.

> **Important**
>
> This article applies to Azure DevOps Services and Azure DevOps Server 2019 and later versions. To customize any project defined on a collection for TFS 2018 or earlier, see **On-premises XML process model**.

With a custom rule, you can define a number of actions based on specific conditions. For example, you can apply a rule to support these types of scenarios:

- When a value is defined for Priority, then make Risk a required field
- When a change is made to the value of Release, then clear the value of "Milestone"
- When a change was made to the value of Remaining Work, then make Completed Work a required field
- When the value of Approved is True, then make Approved By a required field
- When a user story is created, make the following fields required: Priority, Risk, and Effort
- When current user is a member of "Project Administrators", then make Priority required
- When current user is not a member of "Project Administrators", then hide the Priority field

> **Note**
>
> You make a field required and specify a field default through the **Options tab for the field**.

Prior to defining a custom rule, review Rules and rule evaluation, Inherited process rule composition. For examples that illustrate common scenarios for applying rules, see

Sample rule scenarios.

Prerequisites

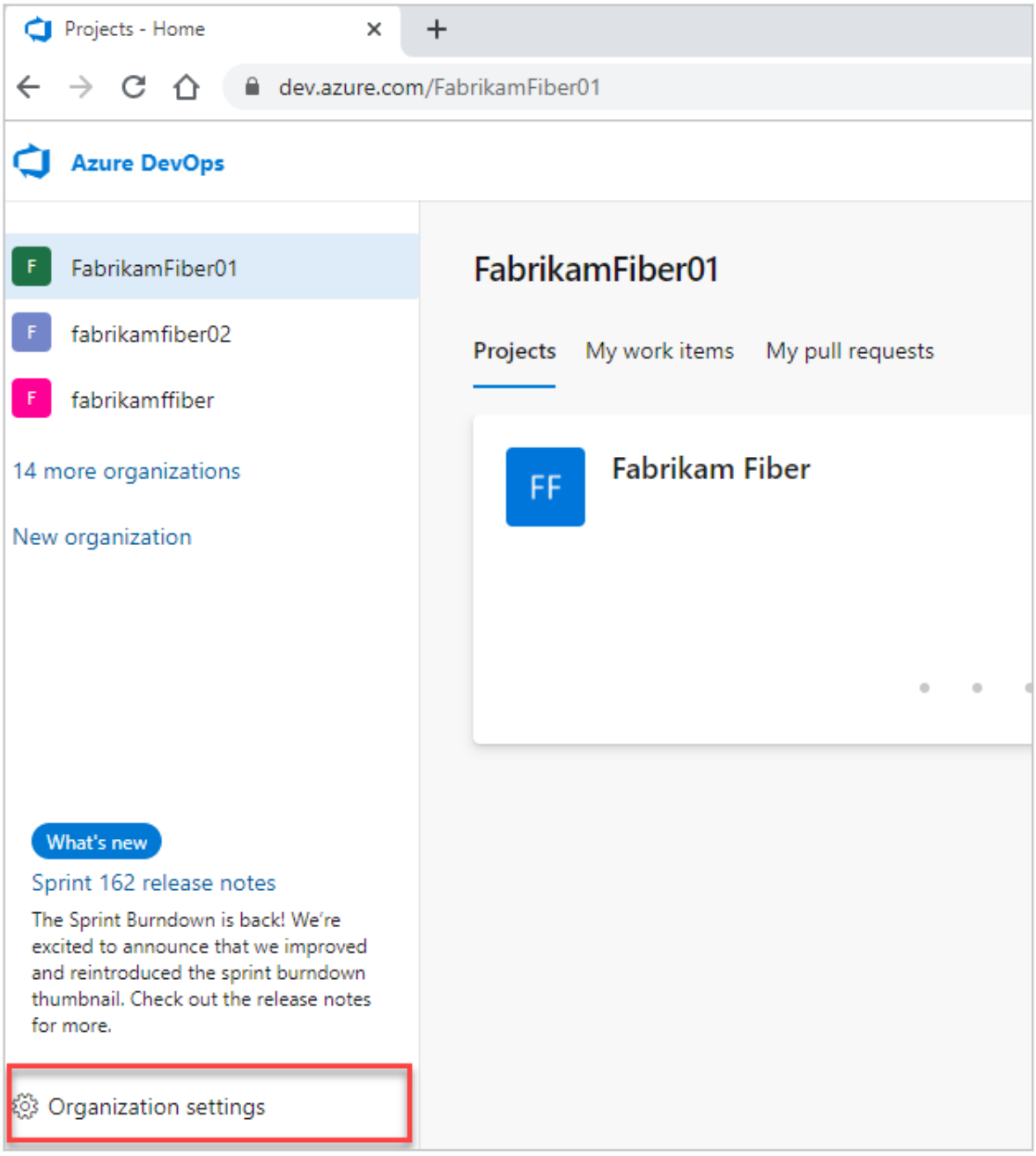
Prior to customizing a process, we recommend that you review Configure and customize Azure Boards, which provides guidance on how to customize Azure Boards to meet your business needs. For a description of the different backlogs and boards, see Tasks supported by Backlogs, Boards, Taskboards, and Plans.

- You must have an organization created in Azure DevOps Services. If you haven't created one yet, do that now.
- To create, edit, and manage processes, you must be a member of the **Project Collection Administrators** group, or have the corresponding collection-level permissions **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**. See Set permissions and access for work tracking, Customize an inherited process.

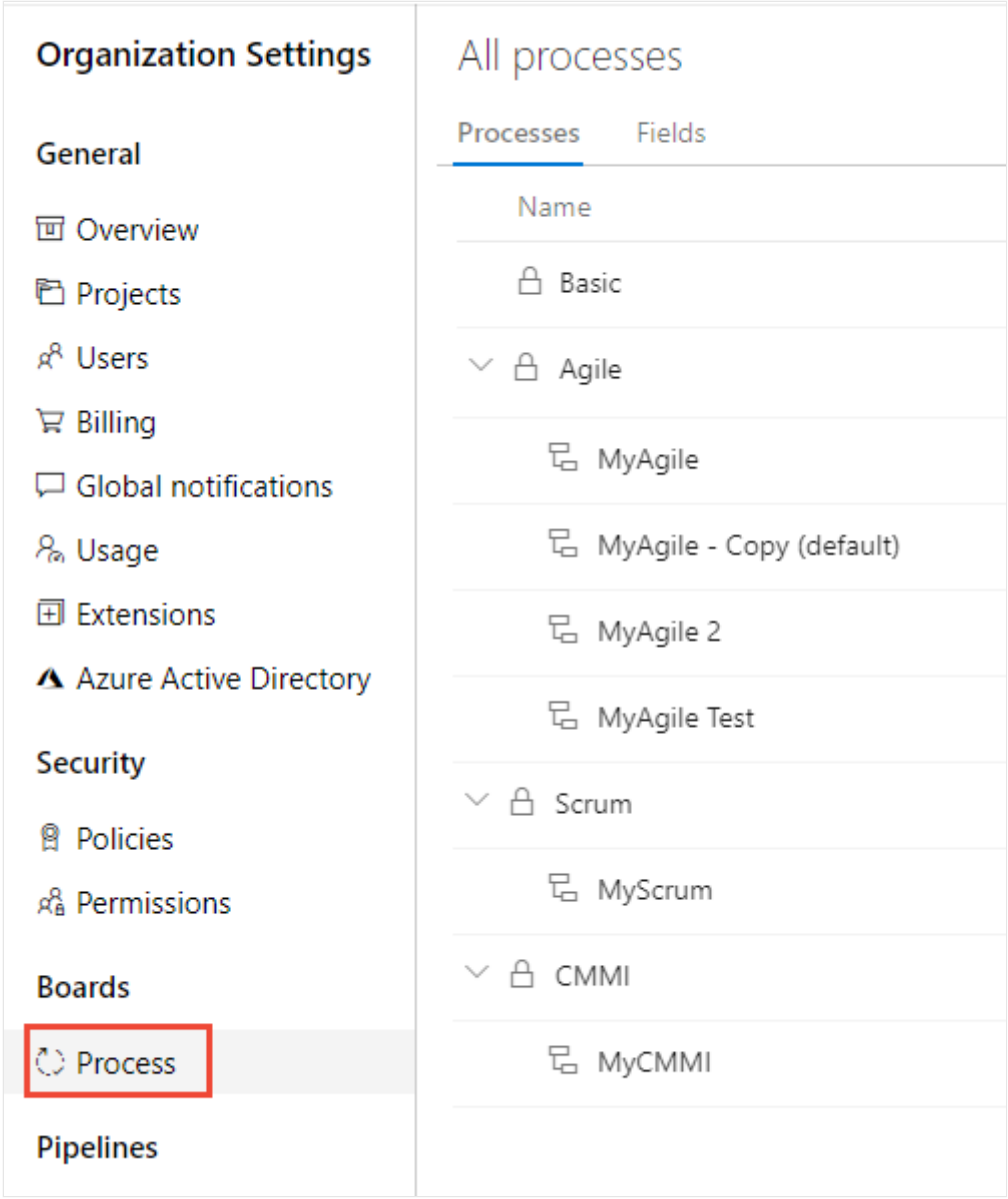
Open Settings>Process

You create, manage, and make customizations to processes from **Organization settings>Process**.

1. Choose the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.



2. Then, choose **Process**.



Important

If you don't see **Process**, then you're working from TFS-2018 or earlier version. The **Process** page isn't supported. You must use the features supported for the **On-premises XML process model**.

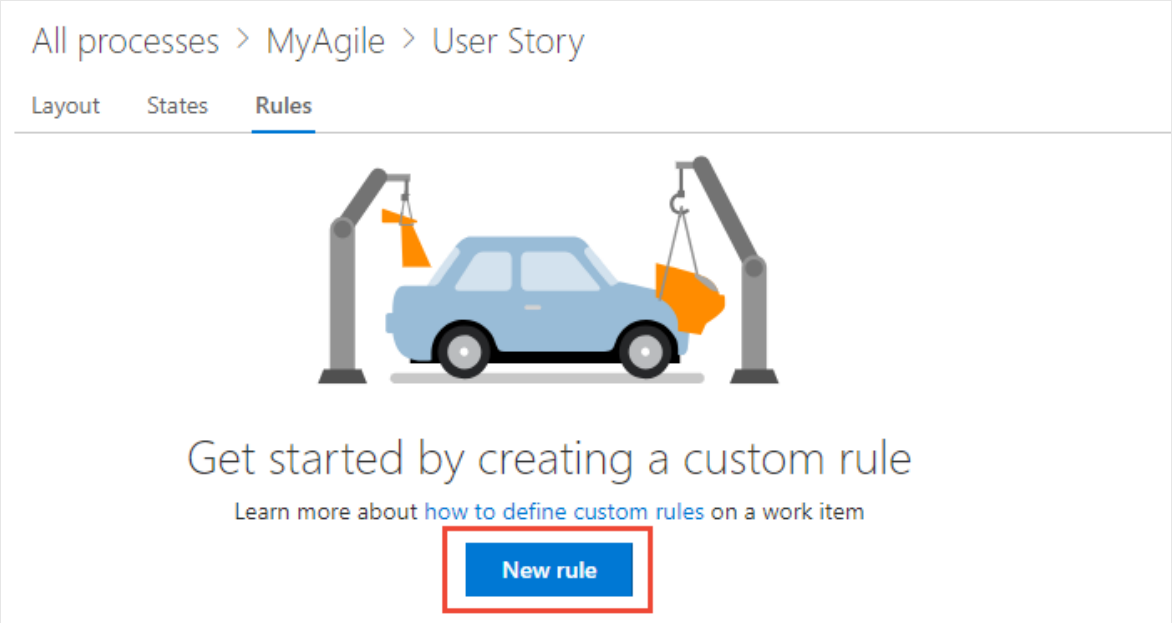
Note

As you customize an inherited process, all projects that use that process are updated automatically to reflect the customizations. For this reason, we recommend that you create a test process and test project when you have a number of customizations to make in order to test the customizations prior to rolling them out to your organization. To learn more, see **Create and manage inherited processes**.

Add a custom rule

You add fields to a selected work item type.

1. Select the WIT to which you want to add a rule, choose **Rules**, and then choose **New rule**.

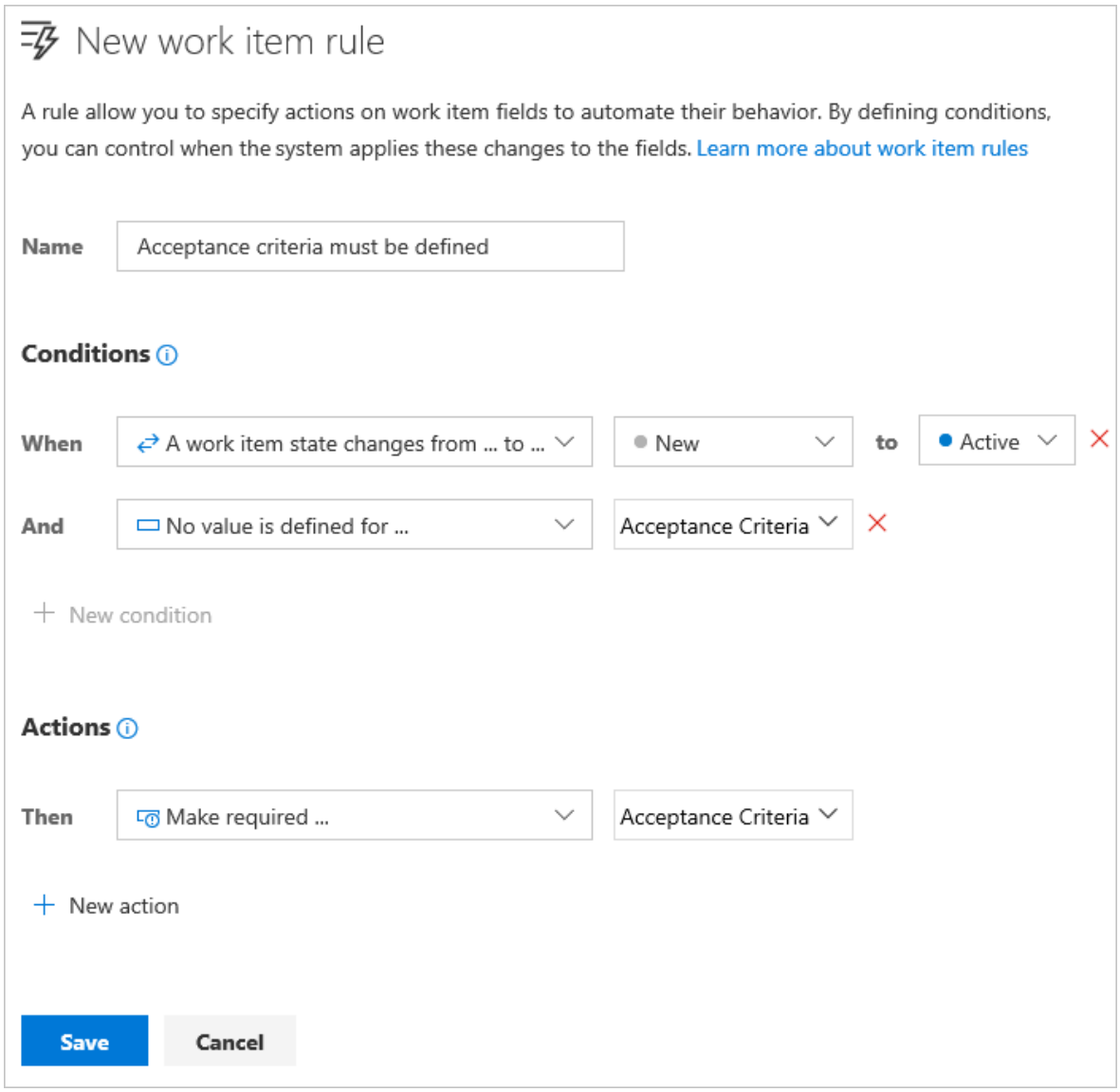


   If you can't fill out the New work item rule dialog, you don't have the necessary permissions to edit the process. See Set permissions and access for work tracking, Customize an inherited process.

2. Name the rule and select the condition(s) and action(s) from the dropdown menus.

   > **Tip**
   >
   > Specify a name that builds off the field(s) you're acting on, or the conditions you're setting.

   Here we define that the **Acceptance Criteria** field is required when the **State** changes to **Active** and it is currently empty.



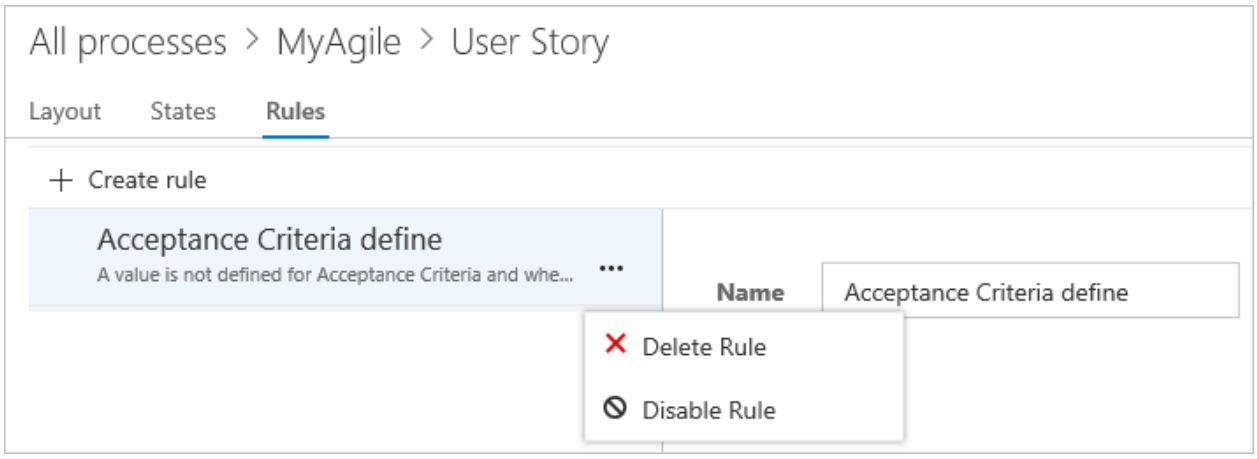
The sequence of actions you specify doesn't impact the behavior of the rule itself or its behavior with respect to other rules defined for the same WIT.

3. Once you've added a custom rule, open a work item and verify that the rule works as you intended.

Delete or disable a rule

You can temporarily disable a rule or delete it altogether.

You delete or disable the rule from the actions menu of the rule.



Related articles

> **Note**
>
> You can review changes made to an inherited process through the audit log. To learn more, see **Access, export, and filter audit logs**.

- Apply rules to workflow states
- Sample rule scenarios
- Rules and rule evaluation
- Customize the web layout
- Customize a project using an inherited process
- Work item form caching

Add and manage fields (Inheritance process)

Article • 12/06/2023

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

You can add a custom field to support tracking additional data requirements or modify select attributes of an  inherited field. For example, you can add a custom field or change the label that appears in the work item form for an inherited field.

For a list of all fields defined for your organization—which includes all fields defined for system and inherited processes—see View work item fields and attributes.

Once you've added a custom field, you can create queries, charts, or Analytics views and Power BI reports to track data related to it.

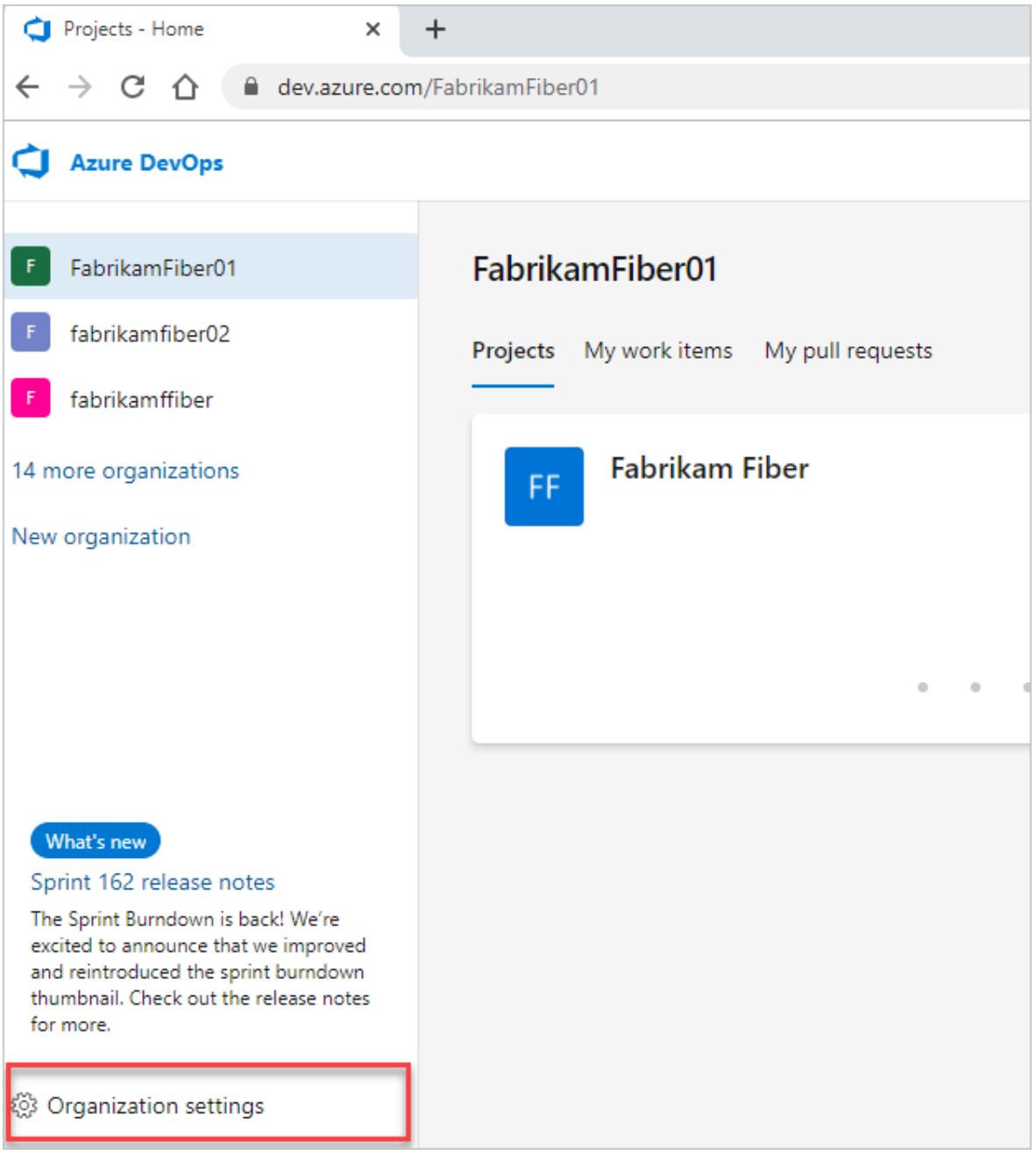
Prerequisites

We recommend checking out Configure and customize Azure Boards, which offers guidance on tailoring Azure Boards to align with your specific business requirements.

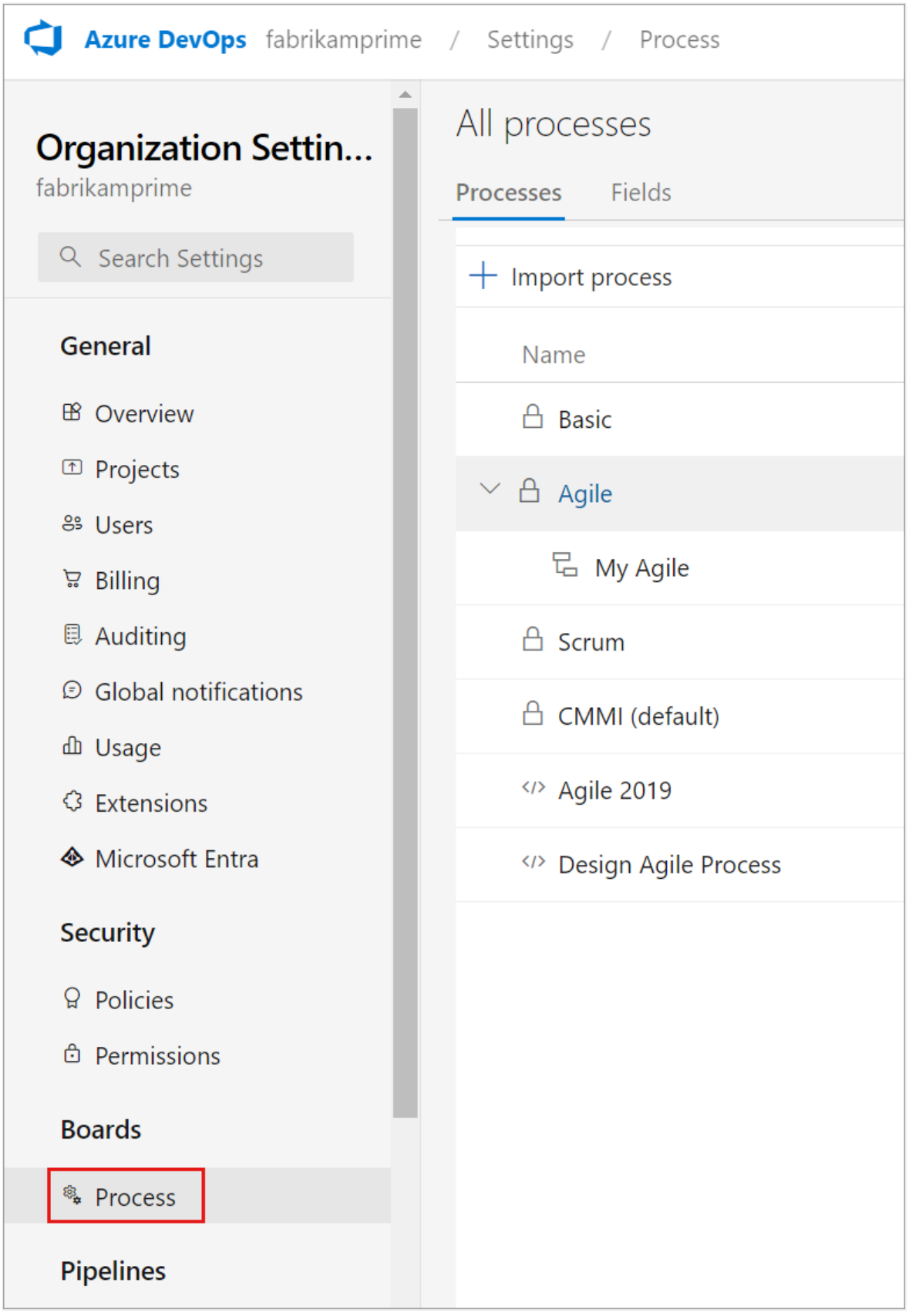
- You must have an organization in Azure DevOps.
- To create, edit, and manage processes, you must be a member of the **Project Collection Administrators** group, or have the corresponding collection-level permissions **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**. For more information, see Set permissions and access for work tracking, Customize an inherited process.

Open Settings > Process

1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`).
2. Select **Organization settings**.



3. Select **Process**.



> **Note**
>
> When you customize an inherited process, any projects using that process automatically reflect the customizations. To ensure a smooth transition, we recommend creating a test process and project, which allows you to test your

customizations before you implement them organization-wide. For more information, see Create and manage inherited processes.

Custom field names

When you add a custom field to an inherited process, Azure DevOps assigns a reference name prefixed with *Custom* and then then name of the field with spaces removed. For example, you add a field named DevOps Triage, the reference name is **Custom.DevOpsTriage**. No spaces are allowed within the reference name.

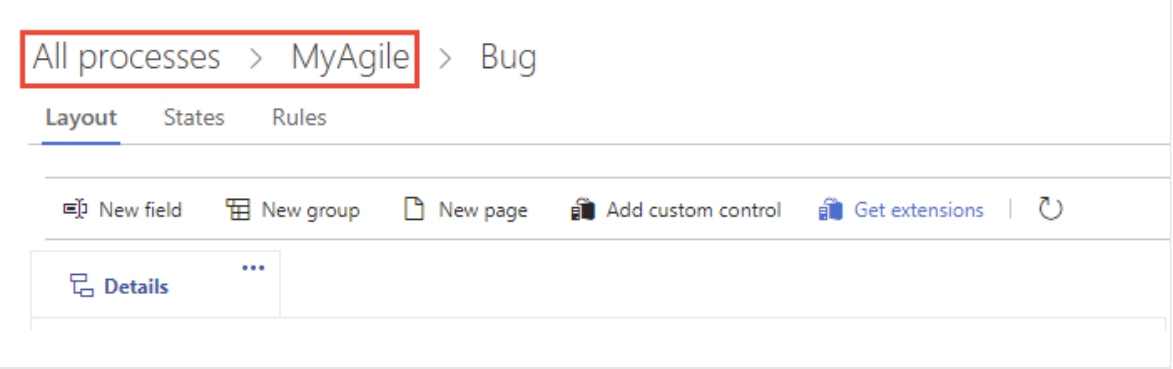
Add a custom field

You can add fields and specify the group and page where they should appear. Also, once you've added a field, you can drag-and-drop it within a page to move its placement within the form. If you have several fields you want to add to a custom page or group, then add those pages or groups first and then add your fields.

> **Note**
>
> Each process can define up to 1024 fields, including system inherited fields. You can only add fields within a page on a form. You can't add a field to the gray area within the form where the Assigned To, State, and Reason fields are located.

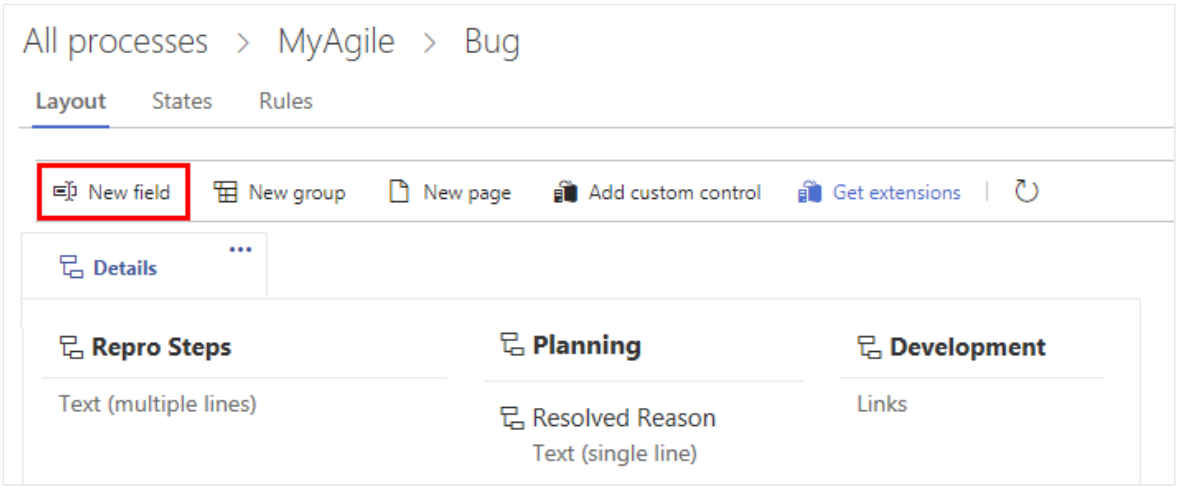
1. From the **Process** page of the selected inherited process, choose the work item type you want to add the custom field to.

   Here we choose the Bug WIT. Note the breadcrumb links that allow you to move back to **All Processes** and **MyAgile** process page.

   If the **New field** and other options are disabled, you don't have the necessary permissions to edit the process. See Set permissions and access for work tracking, Customize an inherited process.

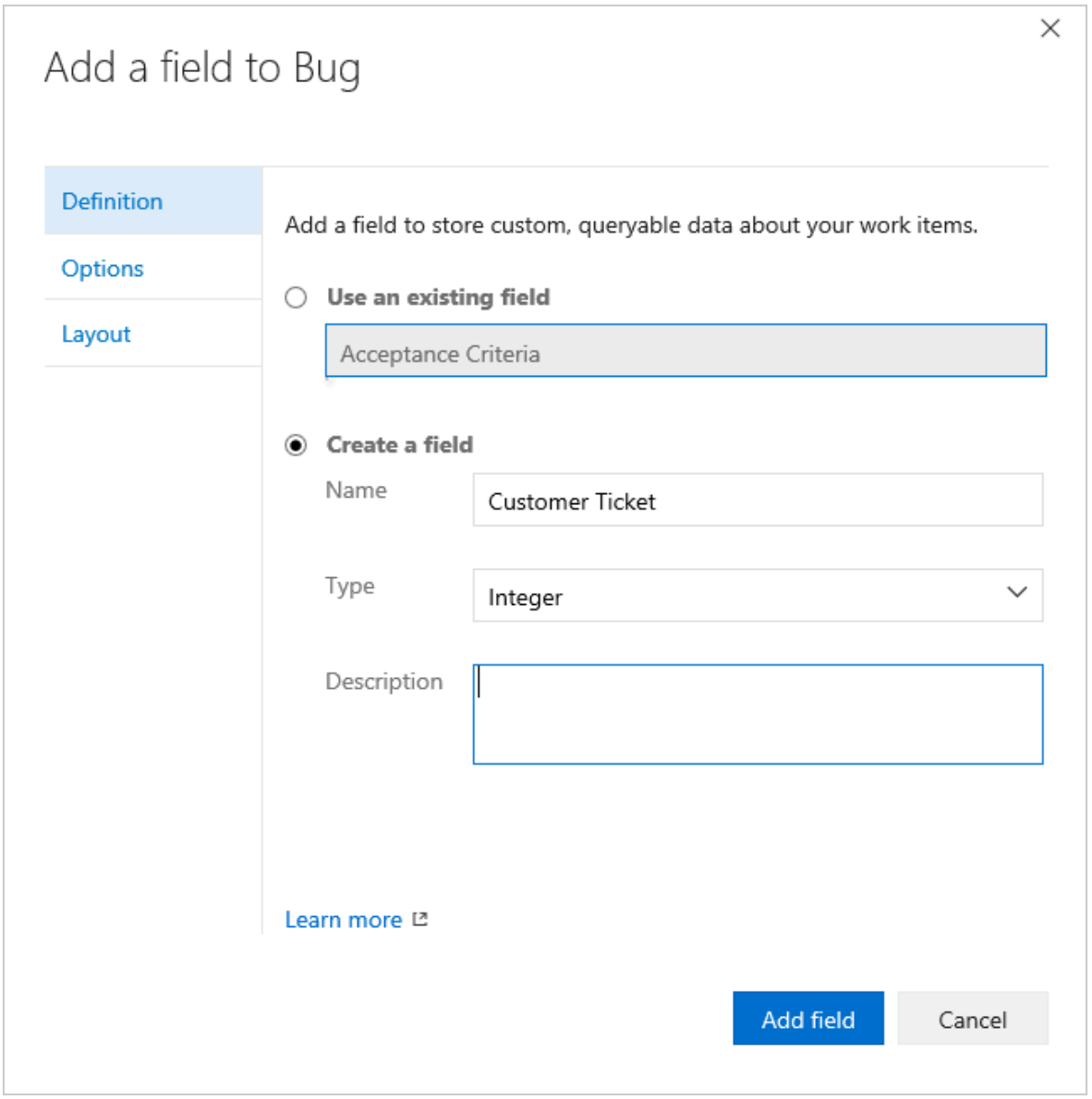
2. With the WIT selected, choose the New field **New field**.



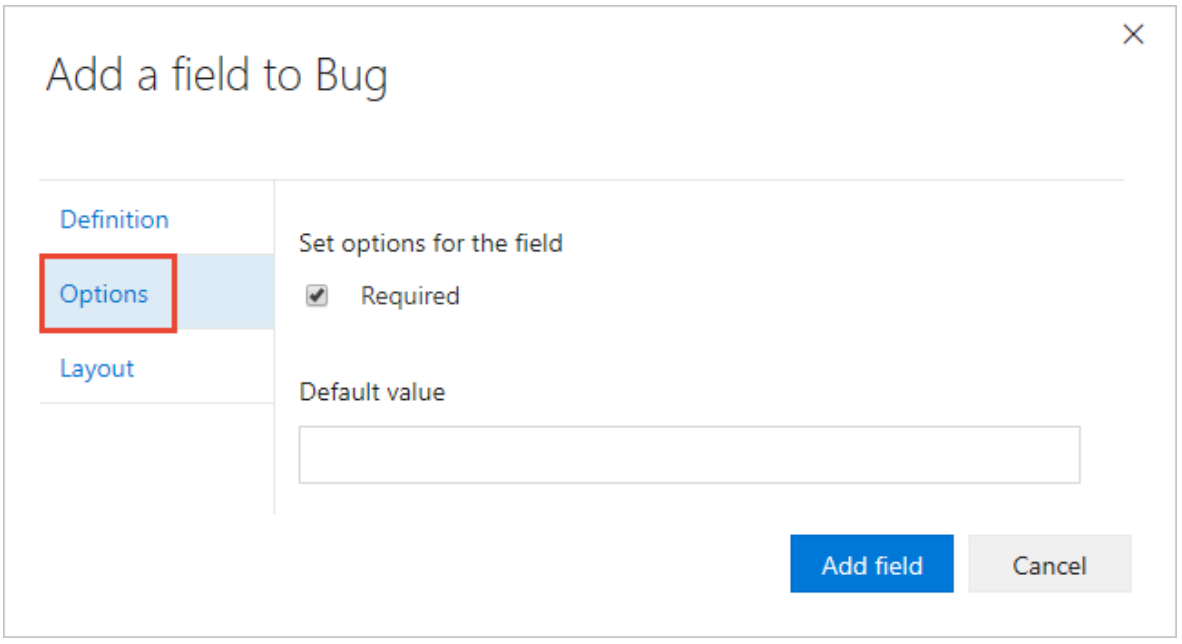
3. Name the field and select the field type from one of the supported data types. Optionally, add a description.

> **Note**
>
> You must specify a field name that is unique within the organization. A custom field defined for one process cannot be the same as the name defined for another process. For more information on field names, see What is a field? How are field names used?

Here we add an Integer field labeled Customer Ticket.

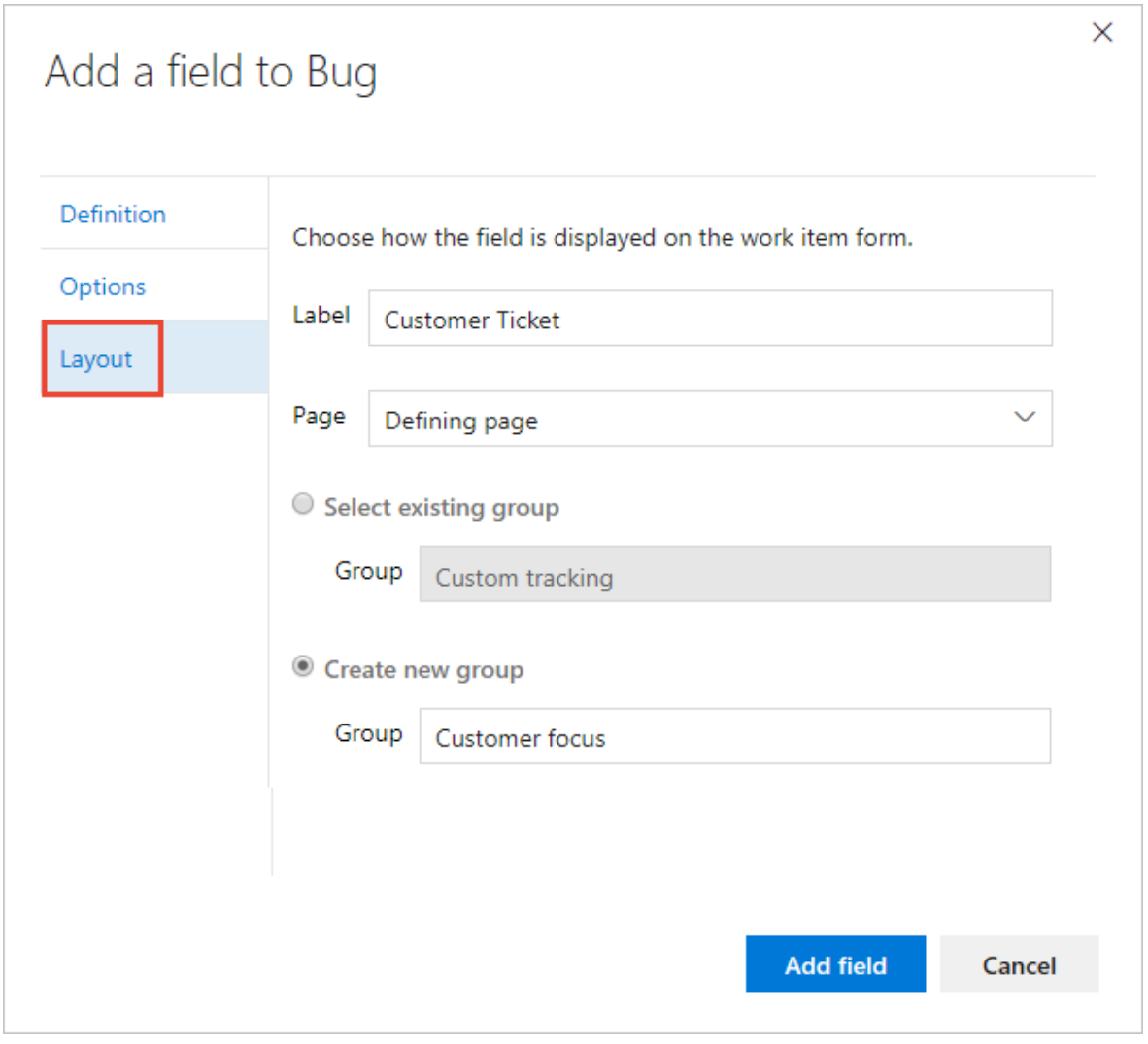


4. (Optional) On the **Options** tab, indicate if the field is required and specify a default value. Or leave the values blank. By making a field **Required**, users must specify a value for the field to save the work item. When you create a work item as well as every time someone opens a work item and the field is empty, the default value is set.



5. (Optional) On the **Layout** tab, you can enter a different form label than the name of the field. Also, you can choose the page and group where the field appears on the form.

   Here, we add the Customer Ticket field to a new group labeled Customer focus.

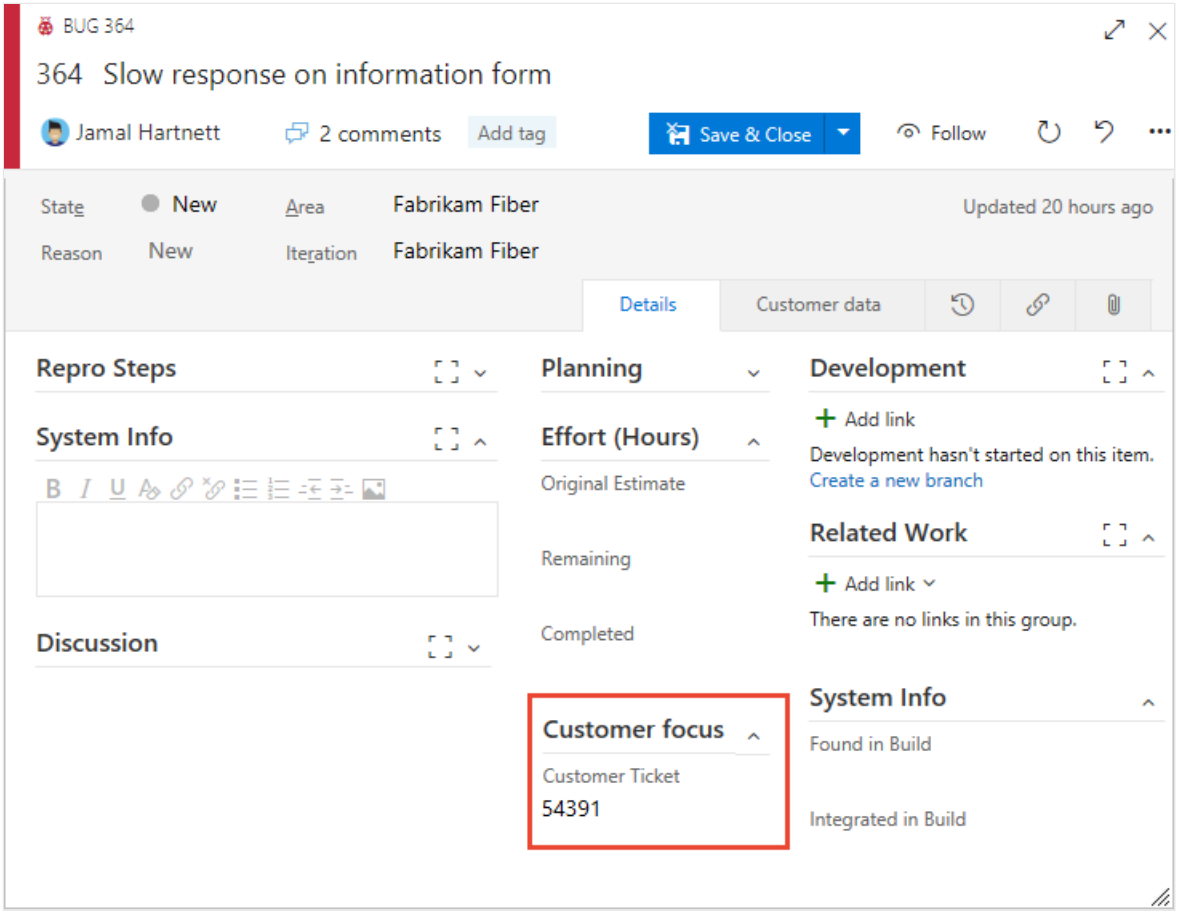


> **Note**
>
> While you can change the form label, you must use the field name when you are adding fields to cards (Kanban, Taskboard) or creating queries based on the field.

6. Choose **Add field** to complete adding the field. If you don't specify its layout location, the system adds it to the first group of fields on the form.

7. When you've completed making all your changes, open a work item of the type you've customized.

   Here, we show the Customer Ticket field has been added to the Status group. You may need to refresh your browser to see the changes.



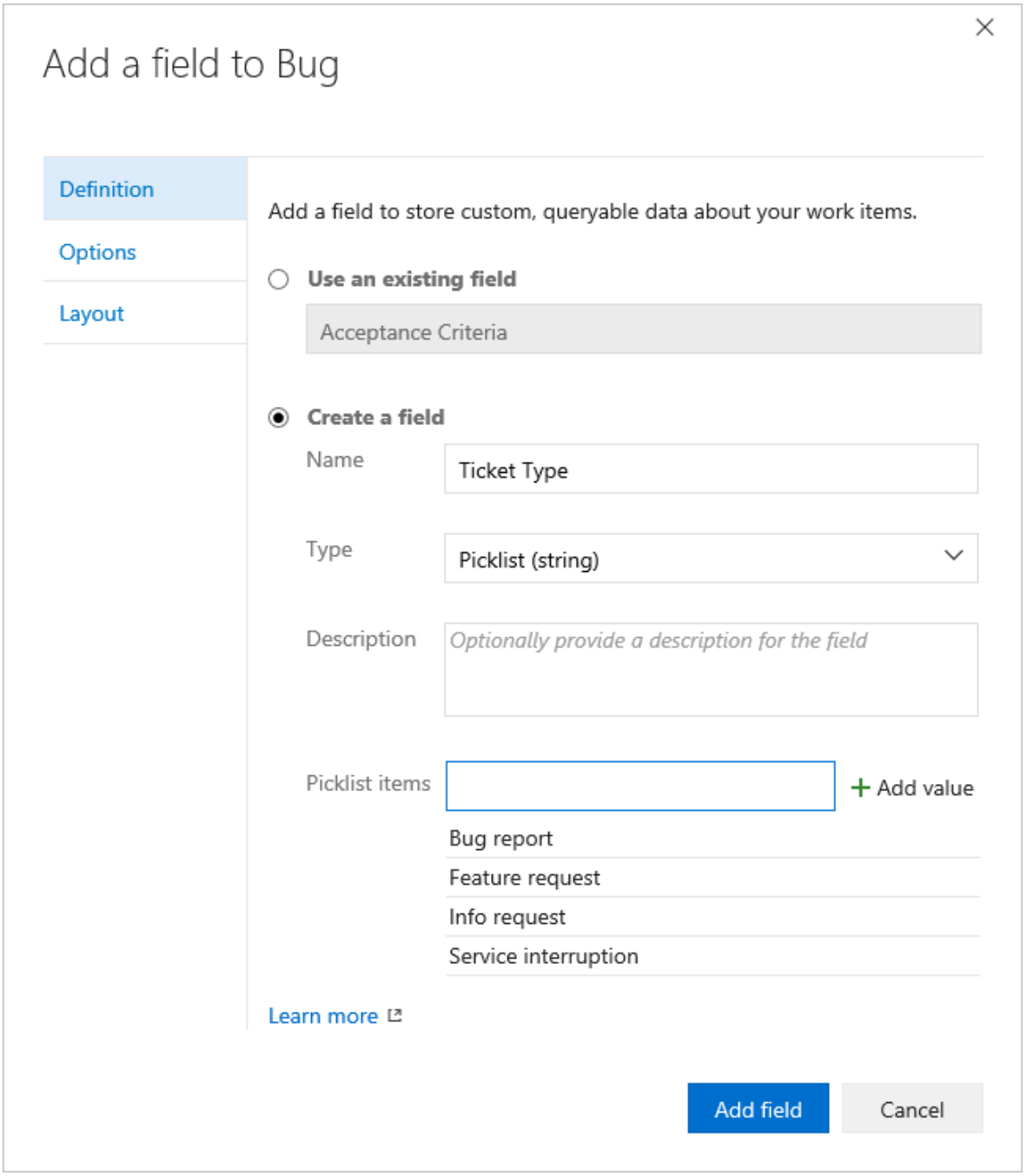
Add a picklist

Work tracking, process, and project limits

You can add a new field and define a pick list or customize the pick list of an inherited field.

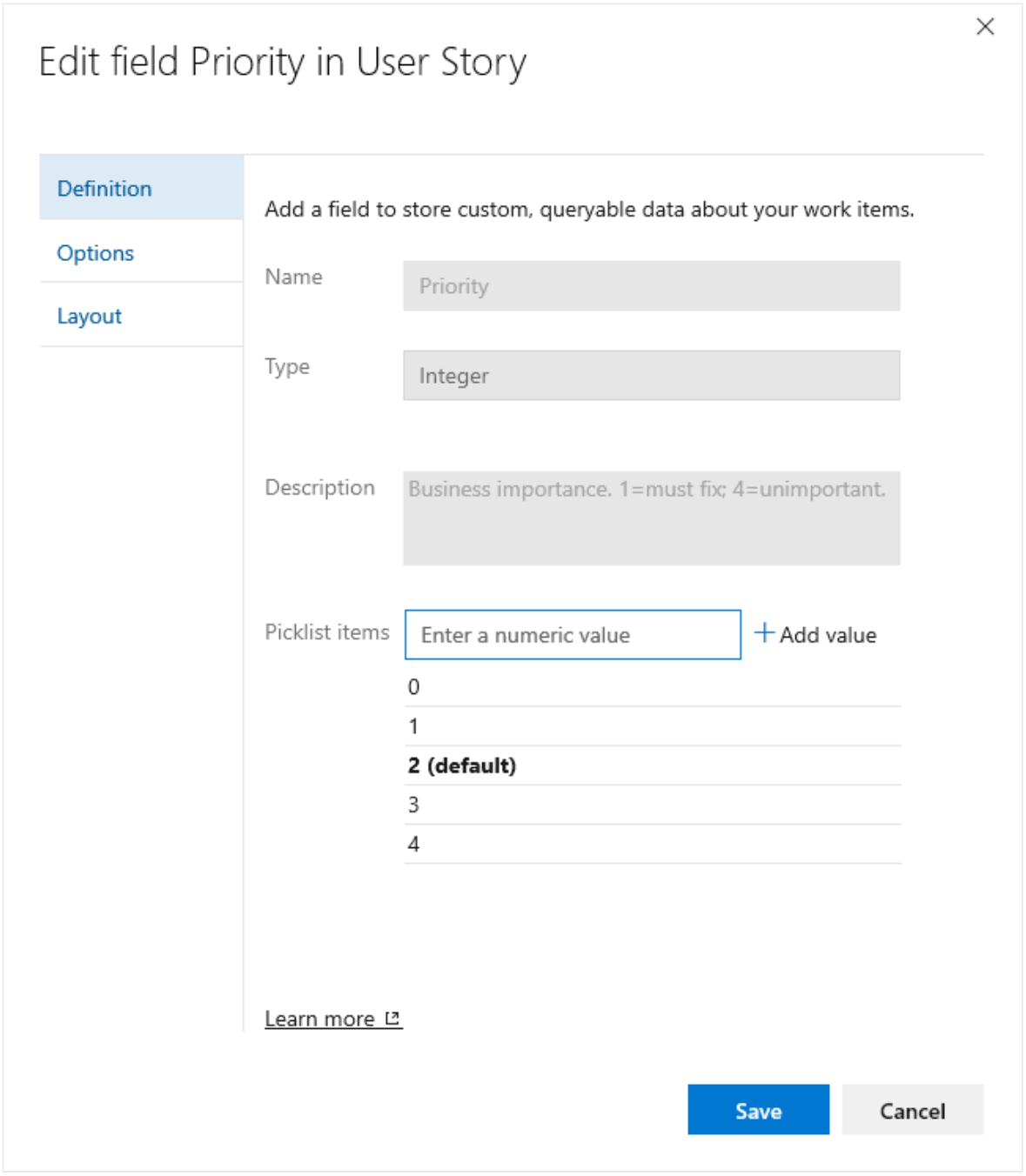
> **Note**
>
> Each organization or collection can define up to 2048 picklists. Each picklist can contain up to 2048 items. Picklist items must be 256 or fewer characters. If you want to add dependent picklists, see Cascading lists.

1. Start by choosing New field **New field**, then specify the picklist type—integer or string—and then add the items to appear in the picklist. You can add an item and then press **Enter** to add another item.

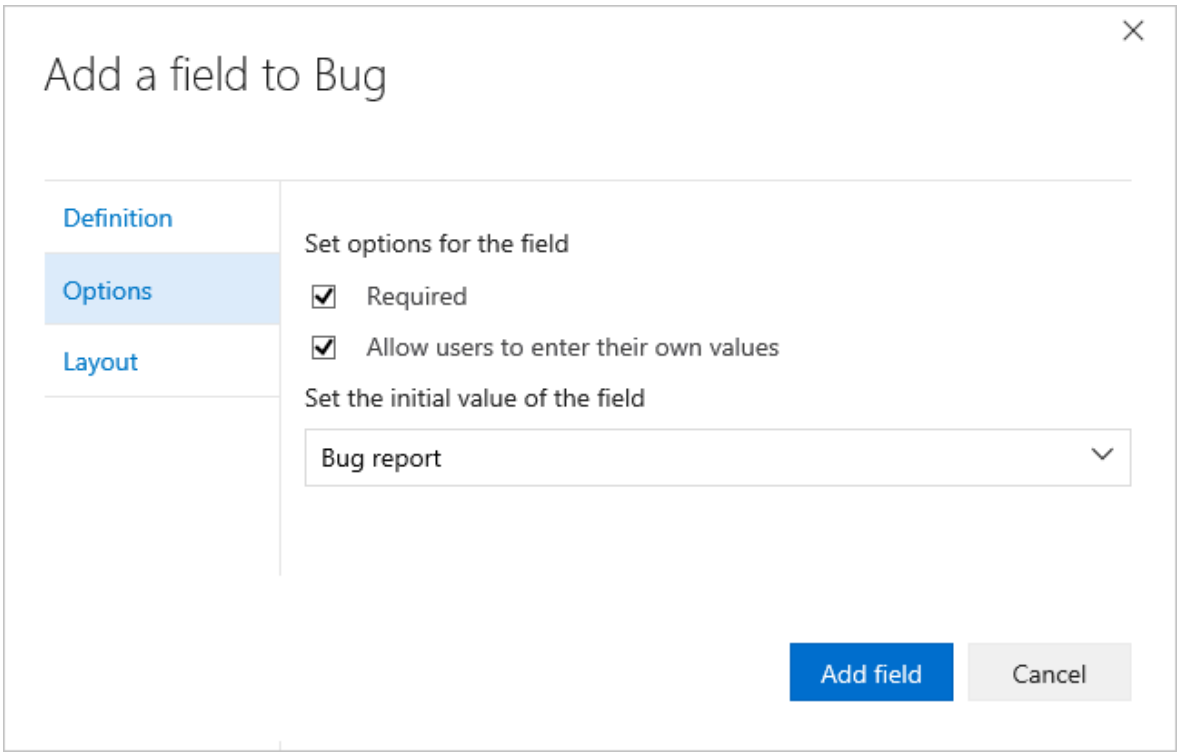


To delete an item in the list, highlight the item and then click the delete icon.

To modify the pick list of an inherited field, choose **Edit** to edit the field. On the **Definition** tab, you can choose to **Add value**.



2. (Optional) Choose the **Options** tab to define the field as required, specify a default, or allow users to enter their own values.

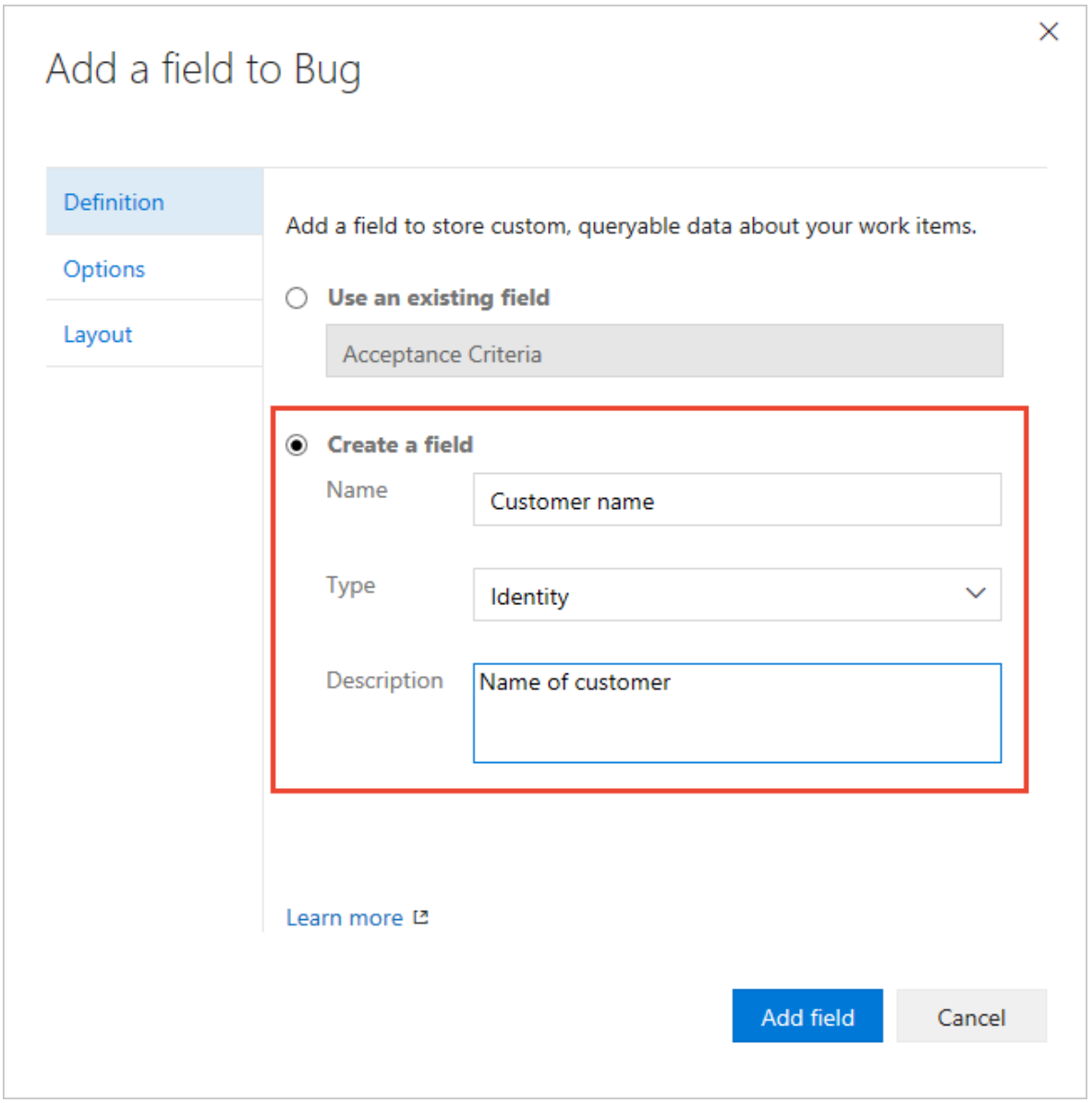


3. (Optional) See previous step 5 to specify where you want the field to appear on the form (**Layout** tab).

Add an Identity field

Use an Identity-based field to add a field similar to the Assigned To field. Identity-based fields act in the same way as the Assigned To field, providing a search and identity picker function. When your organization manages users with Microsoft Entra ID or Active Directory, the system synchronizes Identity-based fields with the names defined in these directories.

1. Start by choosing New field **New field**, then the field name, Identity type, and optionally a description.

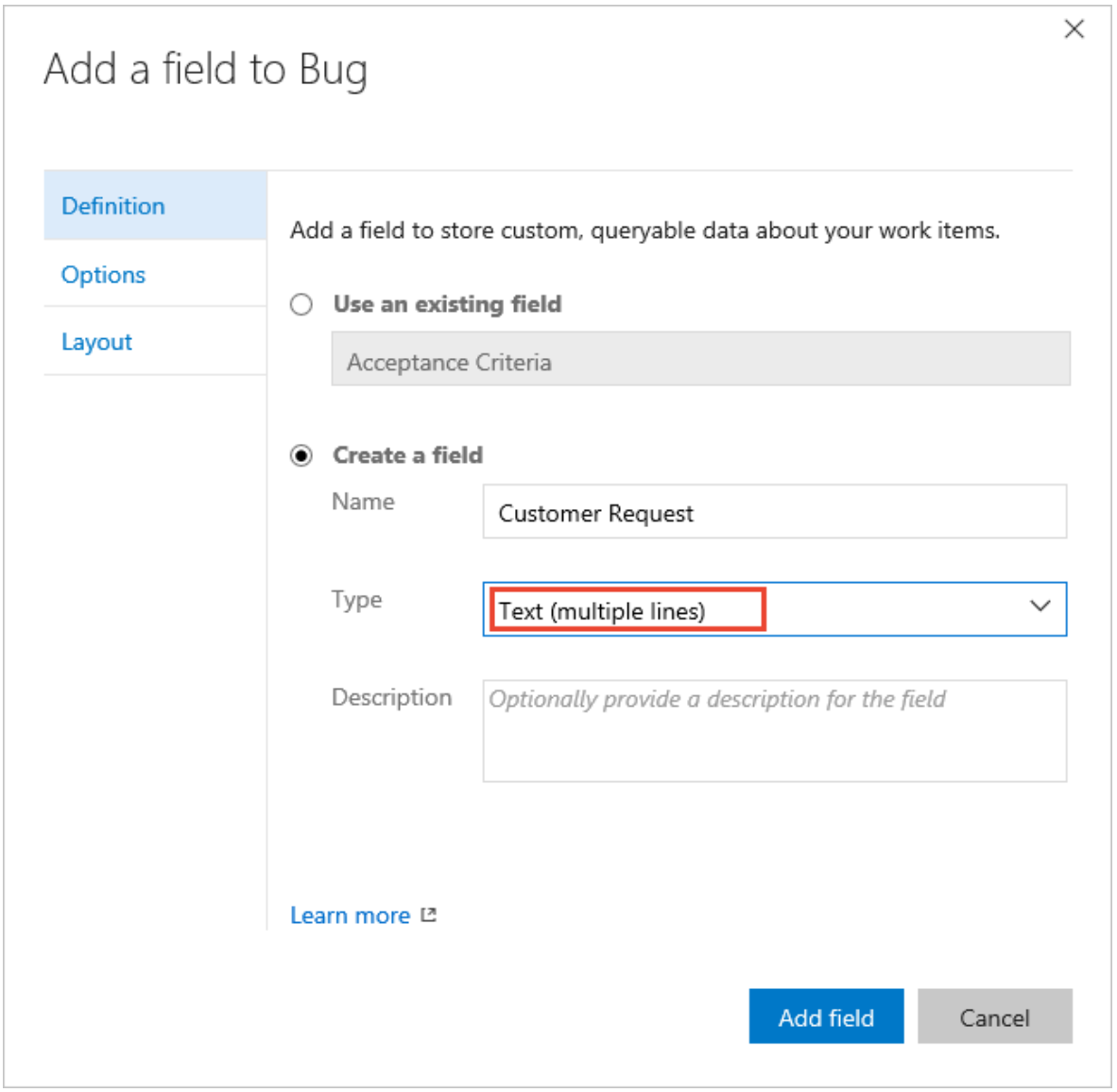


2. (Optional) See previous step 5 to specify where you want the field to appear on the form (Layout tab).

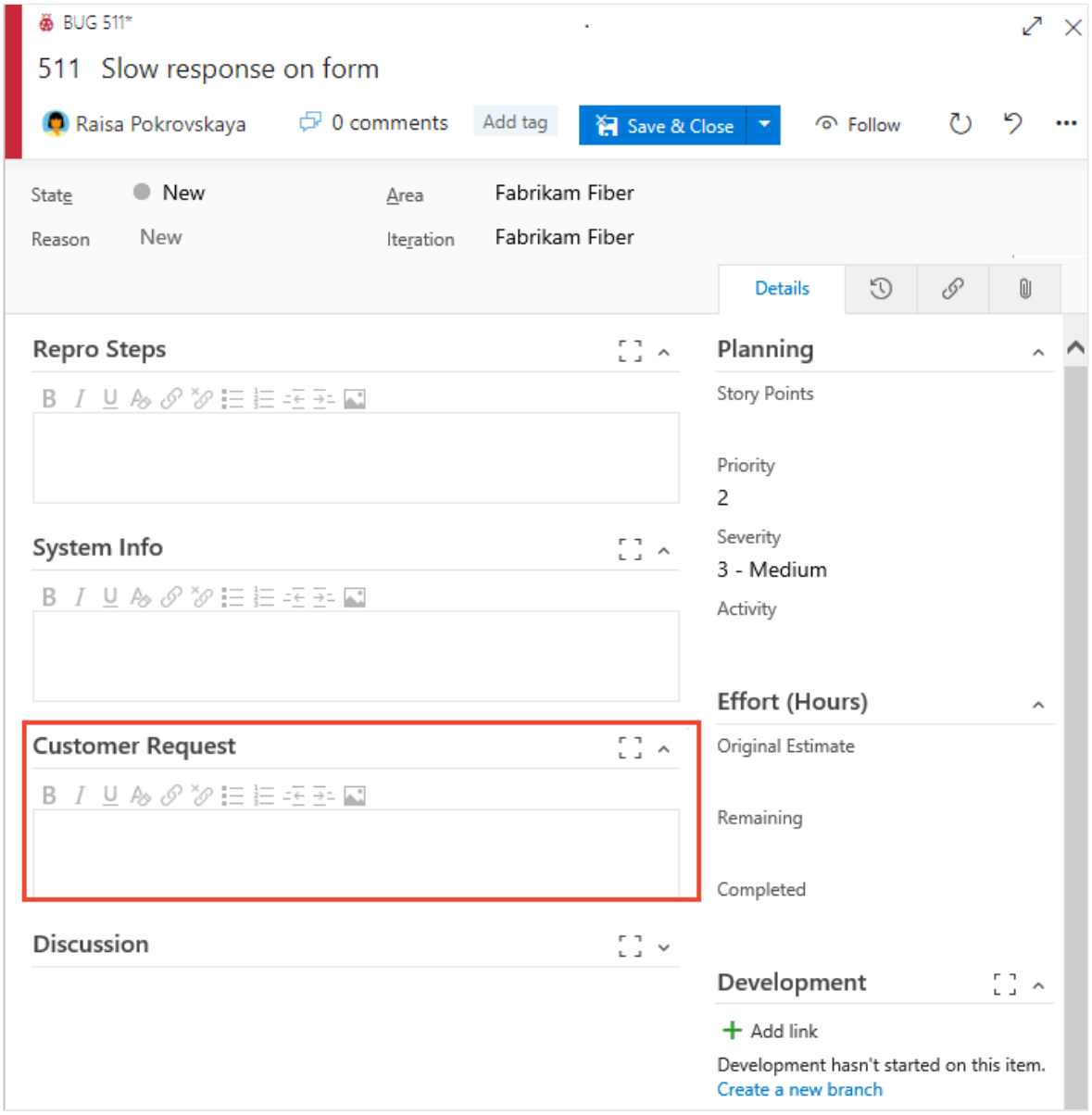
Add a rich-text, HTML field

1. As before, choose the WIT you want to add the field to and then choose the New field **New field**.

2. Choose Text (multiple lines) as the type. Here we label the field as Customer request to capture customer comments for product feature requests.



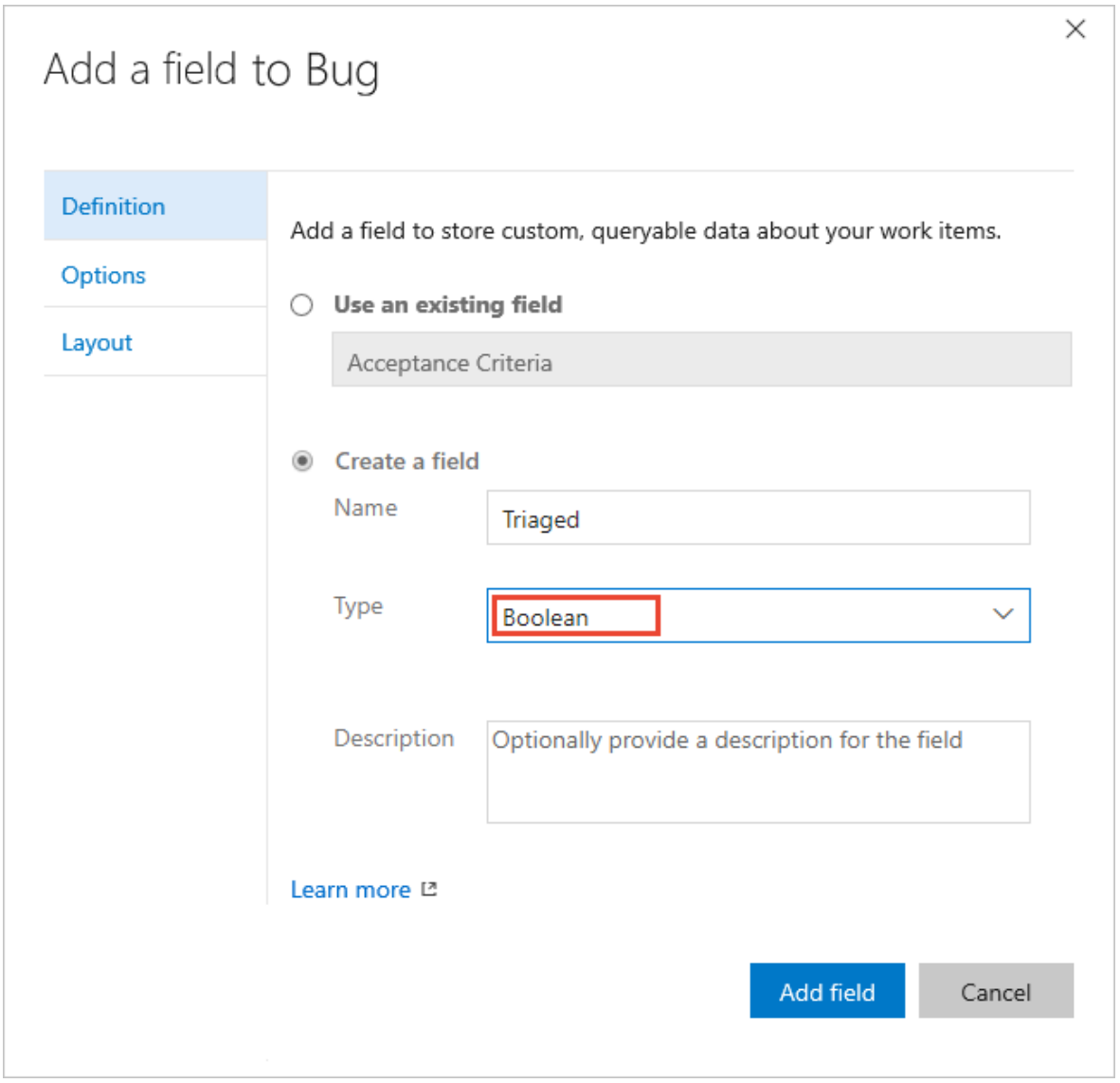
3. The field is added to the first column under all system-defined rich-text fields, but before the Discussion control.



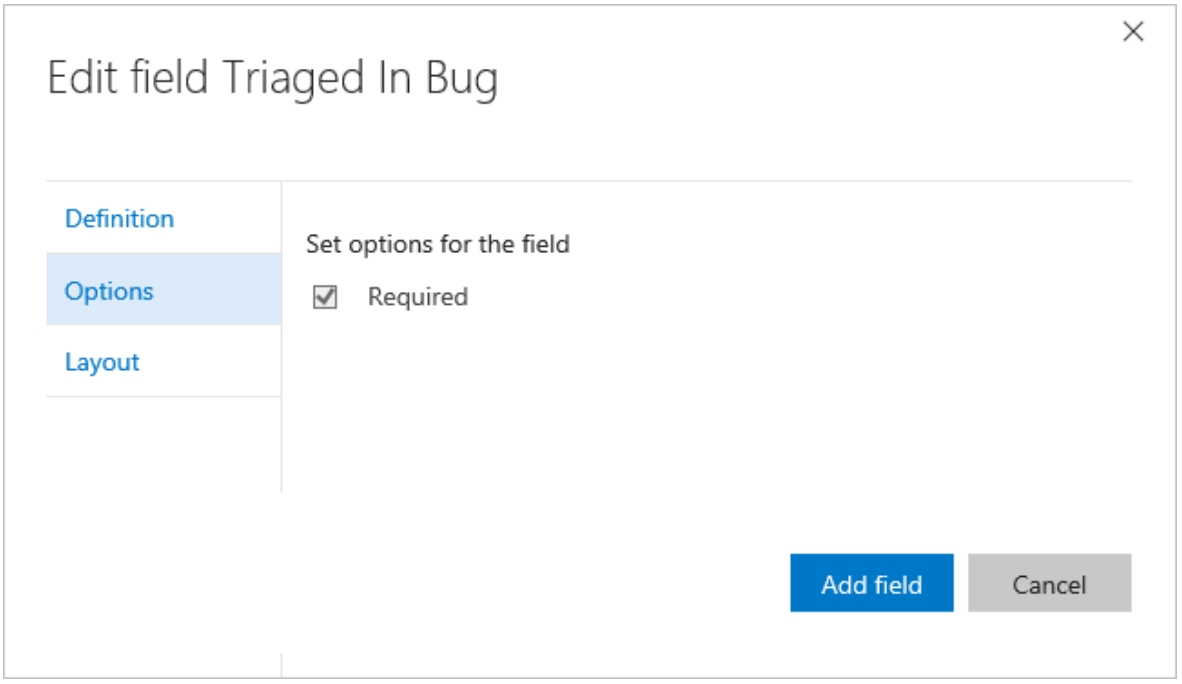
Add a checkbox field

1. As before, choose the WIT you want to add the field to and then choose + **New field**.

2. Choose Boolean as the type, and give it a label. Here we label the field as Triaged to track the triage state of the bug.



3. (Optional) Open the **Options** tab and specify if the field should be required.



4. By default, the field is added to the last group defined in the second column. Open the **Layout** tab to drag and drop the field to another group on the form.

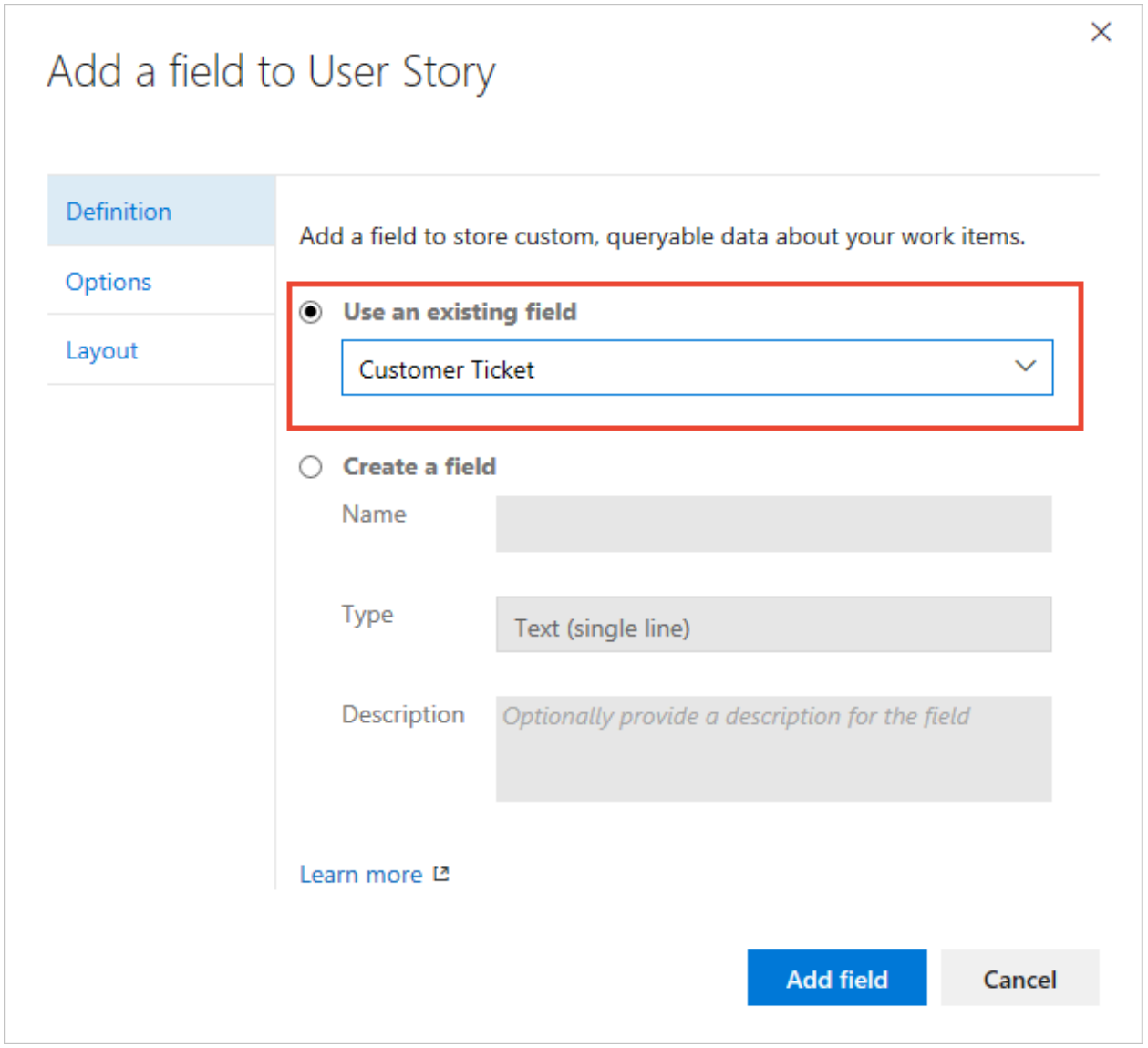
> **Note**
>
> The field appears as a checkbox in the work item form. Check the box to indicate a True value. If you display the field on the Kanban or Taskboard, then the field values of True and False display (not a checkbox).

Add an existing field to another WIT

Existing fields correspond to any inherited field and custom field defined within the collection. Once you've added a custom field to one WIT, you can add it to others from the form menu. Or, you can add a field defined for one process to a work item type in another process. Open the work item type and choose the existing field.

To look up descriptions of any system-defined work item field, see the Work item field index.

Here we add the Customer Ticket field to the User Story WIT.

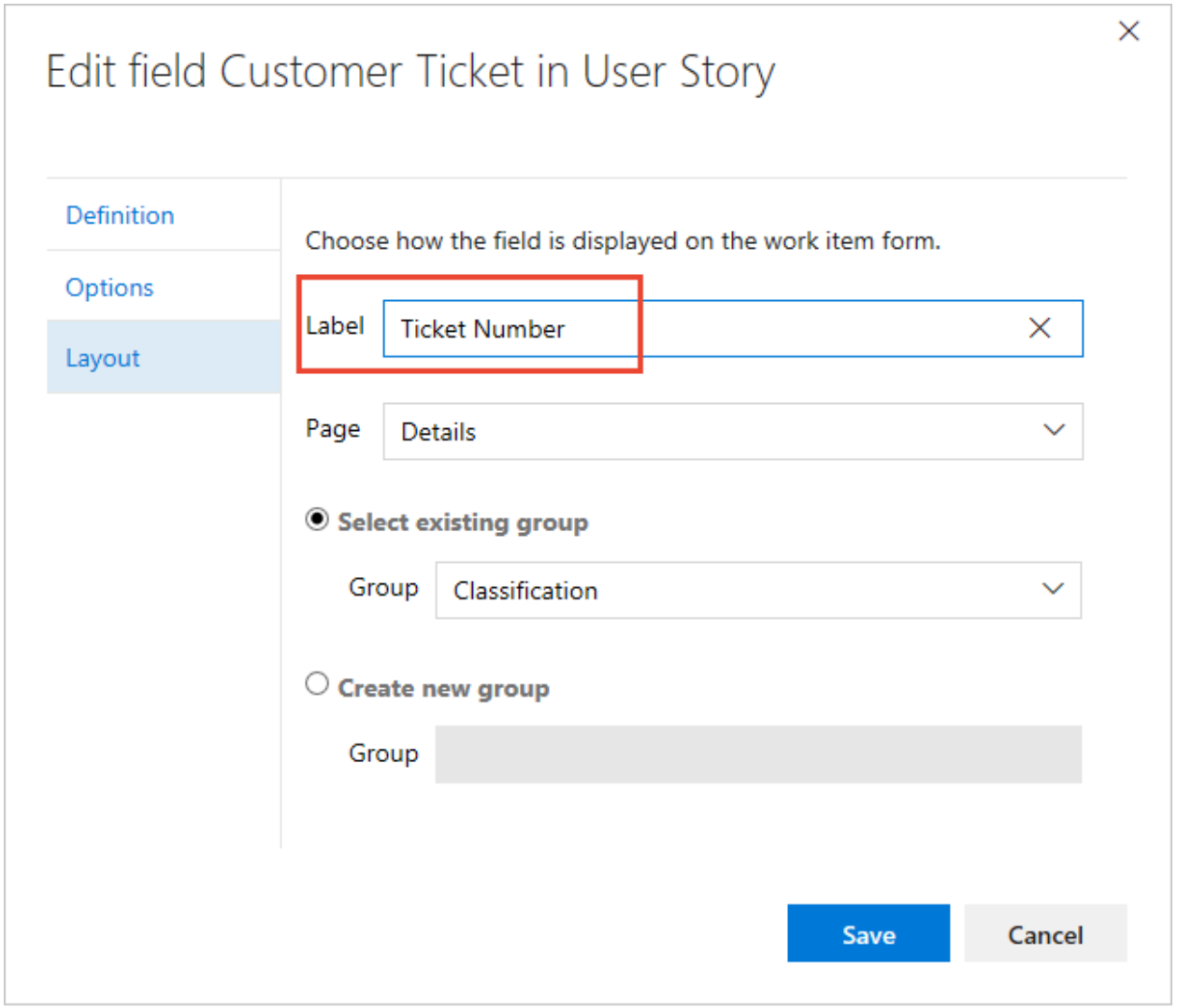


Optionally, specify the Required/Default values and placement within the form for the field.

Relabel a field

Renaming a field or changing the field type aren't supported actions. However, from the **Layout** tab you can change the label that appears on the work item form. When selecting the field in a query, you need to select the field name and not the field label.

Here, we relabel the Customer Ticket field to Ticket Number.



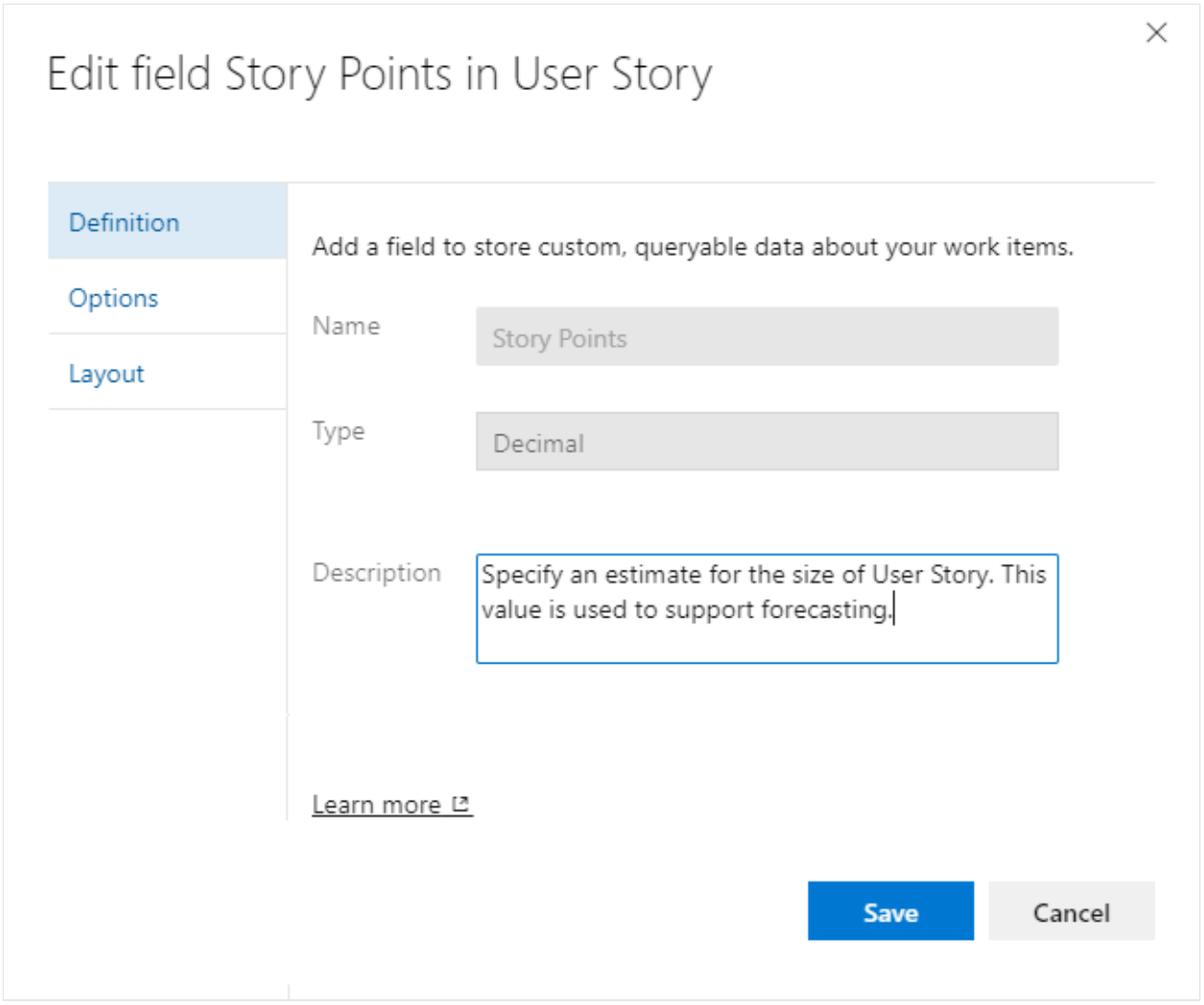
Modify Description help text

Description help text displays when users hover over a field in the work item form. You can define different help text for the same fields for different work item types. Modification is supported for both custom fields and inherited fields. Note that the implementation differs based on the field type.

- **For inherited fields**: Description help text can be defined differently for each work item type and process.
- **For custom fields**: Description help text appears the same for the field across all work item types and all processes.

To modify the **Description** help text, choose the work item type you want to modify, choose **Edit** for the field and choose the **Definition** tab. The modified value only affects that field in the process and for that work item type.

Here we modify the Story Points field for User Story.



Show, hide, or remove a field

You can choose to show or hide any field or custom control from appearing on a form. If you want to reinstate a field onto the form later, you can un-hide These actions differ from the Delete option, which deletes the field from the organization.

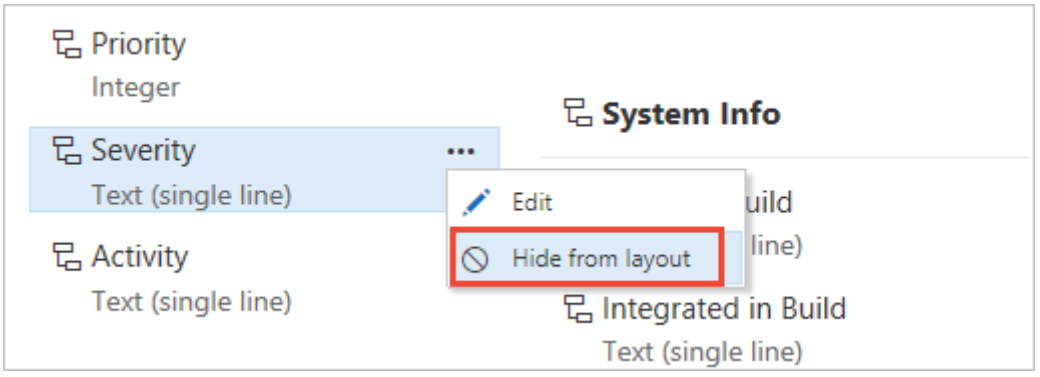
> **Note**

Data defined for an inherited field, even if you hide it, is maintained in the data store and work item history. You can view a record of it by viewing the history tab for a work item.

When you remove a custom field from the layout, it is maintained in the data store but stripped from the history. You can view it from the query results. If you add the field back to the form, then the history for it is restored.To delete a custom field from a project collection, see Delete a field.

Hide a field or custom control

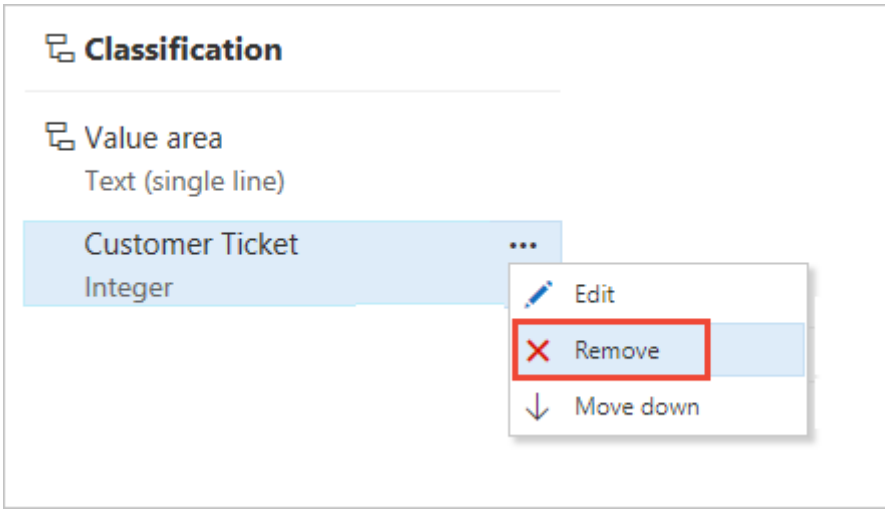
1. Open the context menu for the field or control and choose **Hide from layout**.



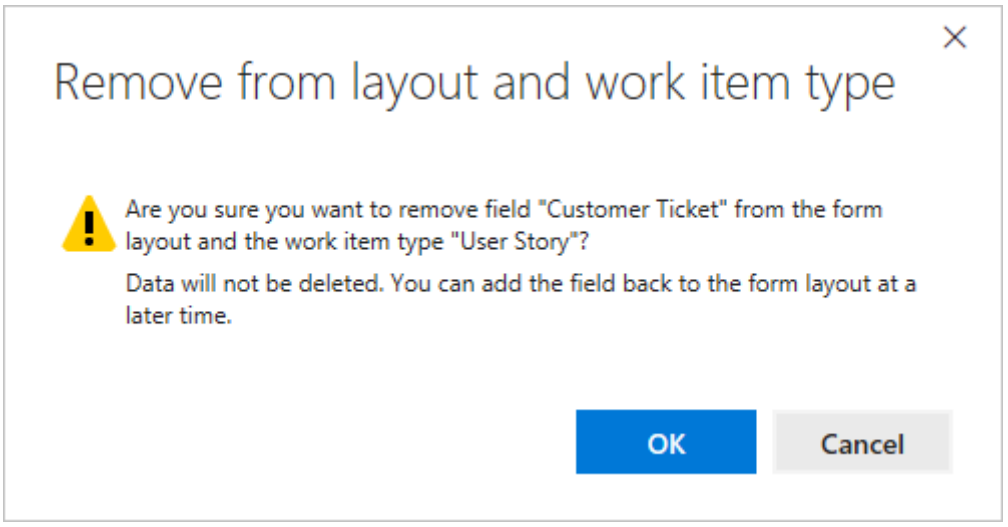
2. To add a hidden field or control to the form, choose **Show on layout**.

Remove a custom field from a form

1. Choose **Remove** from the context menu of the field you want to remove.



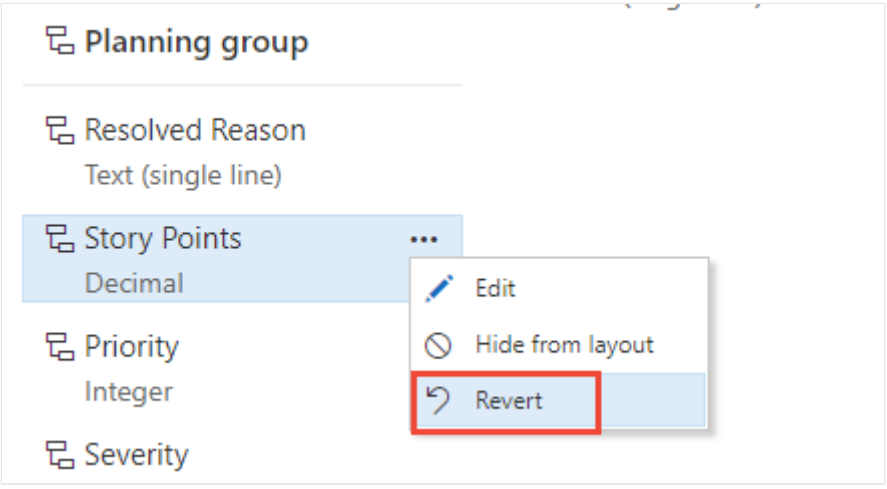
2. Confirm that you want to remove the field.



3. To add a custom field that's been removed, choose **New field** and select **Use an existing field**.

Revert field to preset defaults

You may discard changes you made to an inherited field. From the **Layout** page of the modified work item type, choose the **Revert** option for the field.



Delete a custom field

With the Inheritance process model, you can only delete custom fields. You can't delete fields defined for system default processes.

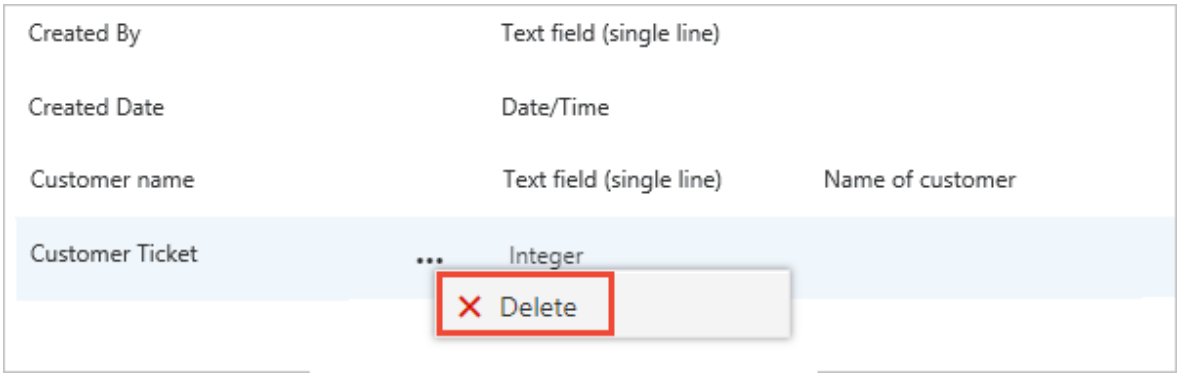
Deleting a field deletes all data associated with that field, including historical values. Once deleted, you can only restore the field and recover the data using the Fields - Update REST API.

> ⓘ Note

We recommend that you only delete fields that are not in use. You can determine fields that aren't in use by exercising the **witadmin listfields** command. To learn how, see Manage work item fields (witadmin).
Or, if Analytics is enabled for your organization or collection, you can query Analytics for where a custom field is in use with the following syntax:

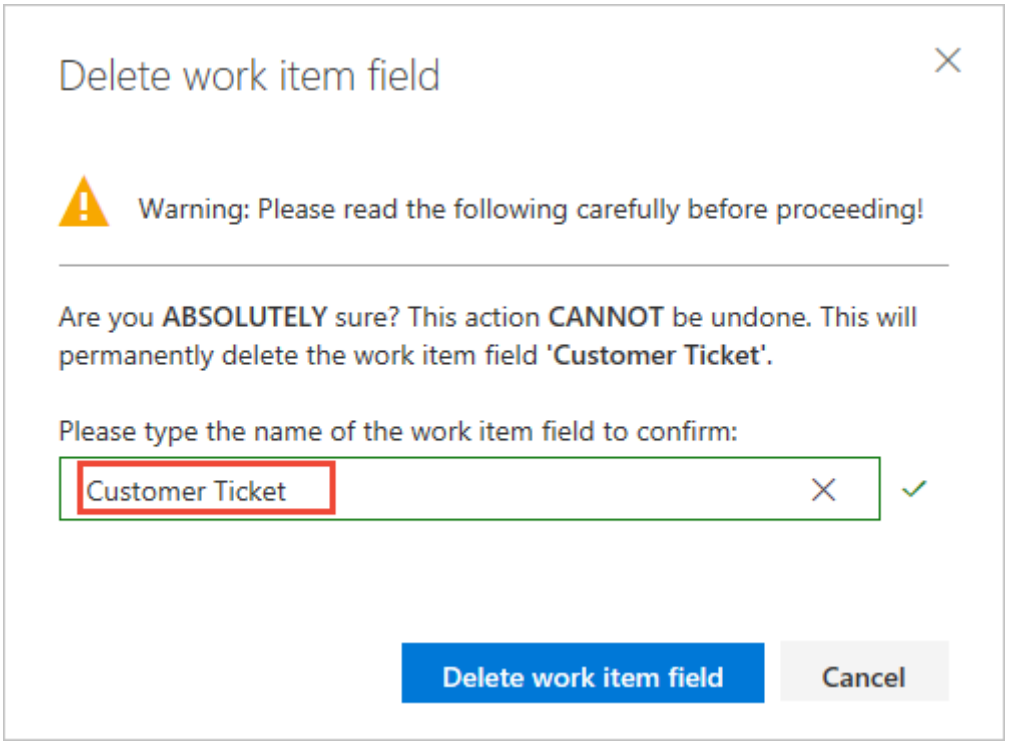
```
https://analytics.dev.azure.com/{OrganizationName}/_odata/v4.0-preview/WorkItemTypeFields?$filter=FieldReferenceName eq {CustomFieldReferenceName}&$select=WorkItemType
```

1. You delete the field from the **All processes**>**Fields** page.



   To delete fields, you must be a member of the Project Collection Administrators group or be granted explicit permissions to Delete field from account.

2. To complete the delete operation, you must type the name of the field as shown.



Related articles

> **Note**
>
> Review changes made to an inherited process through the audit log. For more information, see Access, export, and filter audit logs.

- View work item fields and attributes
- Add or modify a custom work item type
- Customize the web layout
- Customize a project using an inherited process
- Customize cards on boards
- Show bugs on backlogs and boards
- Work tracking, process, and project limits

If you're working with test plans or test cases, see these articles:

- Create a test plan
- Create manual test cases

Feedback

Was this page helpful? Yes No

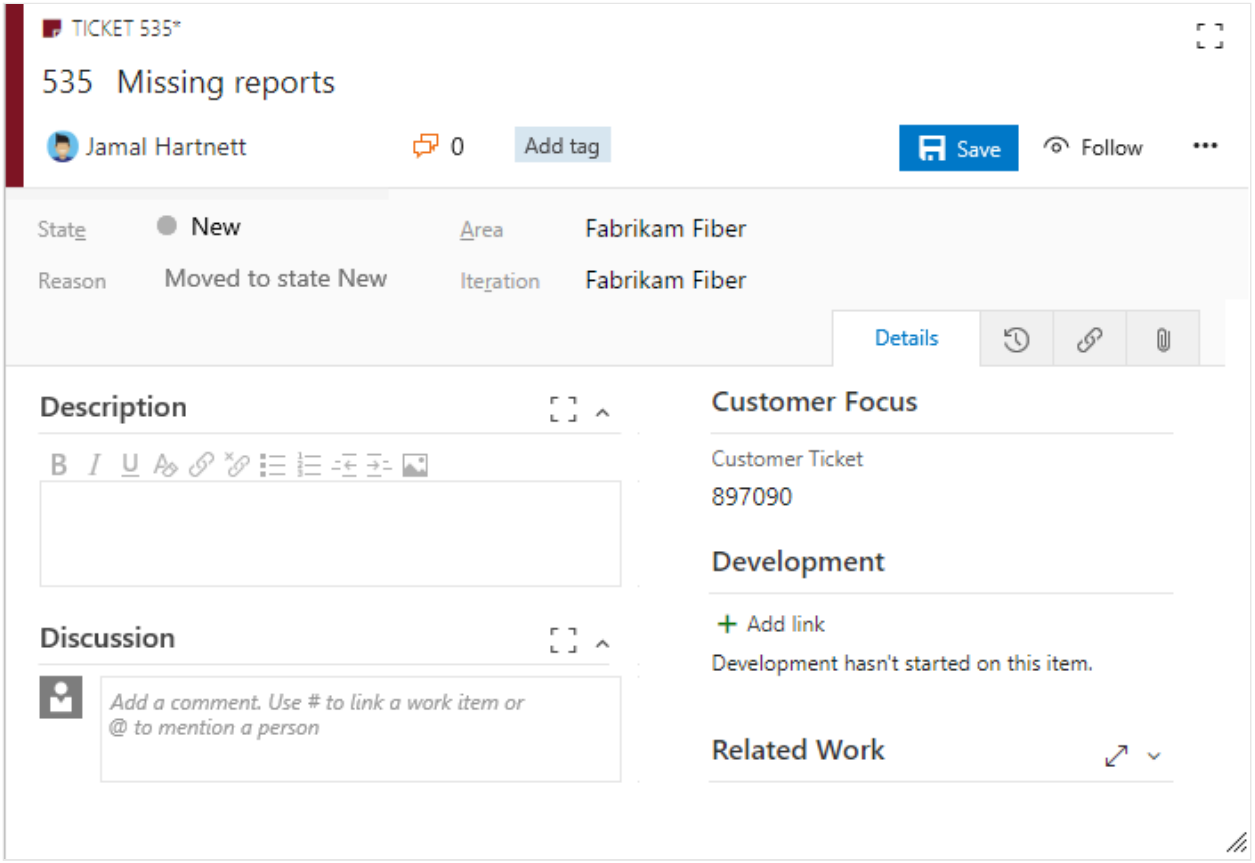
Provide product feedback

Add and manage work item types

Article • 09/24/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

You can add custom work item types (WITs) or modify existing WITs to add fields, remove or hide fields, add custom rules, and more. For example, you may want to capture customer issues in a custom Ticket WIT.



Prerequisites

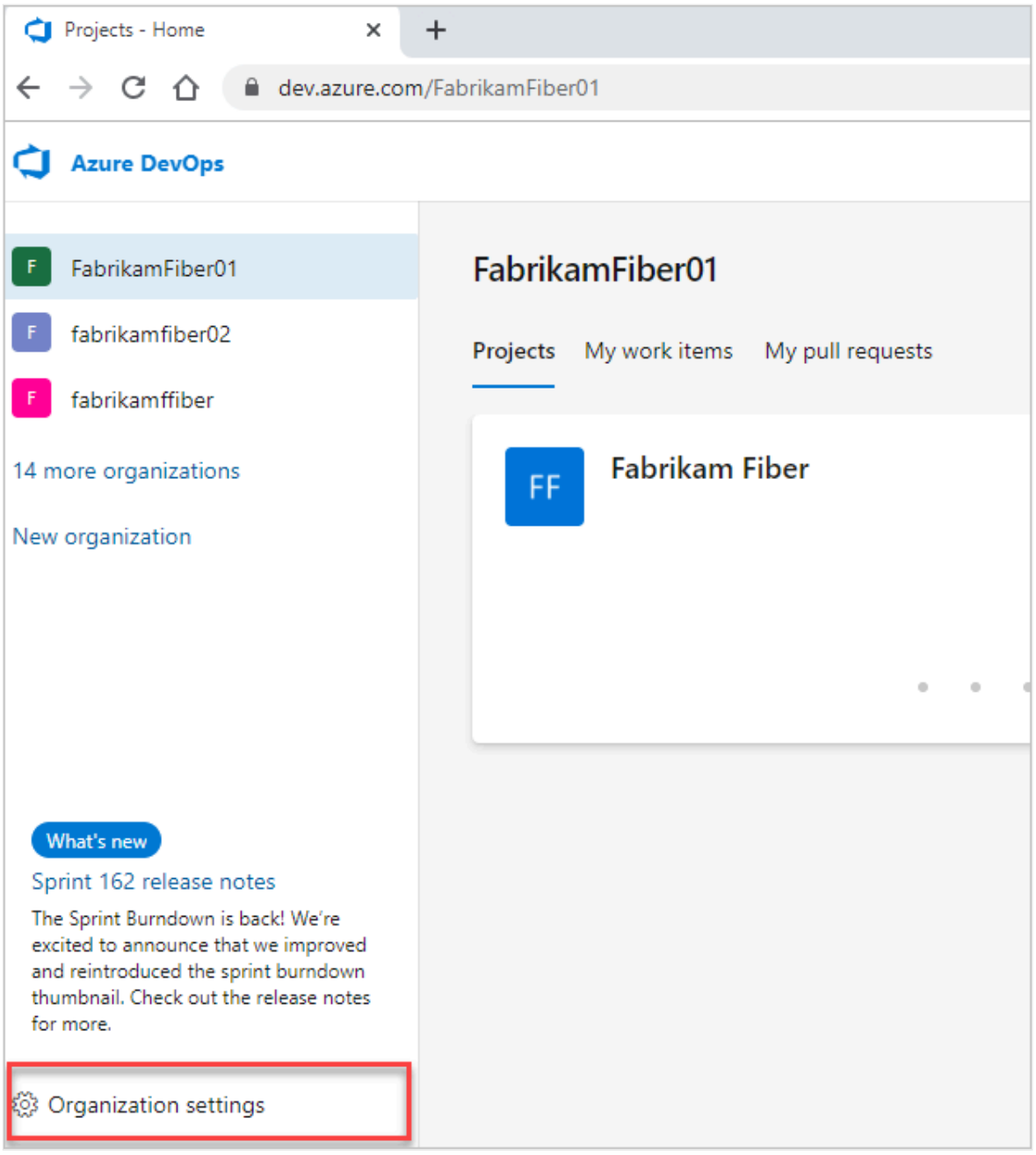
Check out Configure and customize Azure Boards, which offers guidance on tailoring Azure Boards to align with your specific business requirements.

- **Organization requirement**: Ensure you have an organization in Azure DevOps.
- **Permissions**:
  - Be a member of the **Project Collection Administrators** group.
  - Have collection-level permissions such as **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**.
  - These permissions allow you to modify processes and fields within your organization.

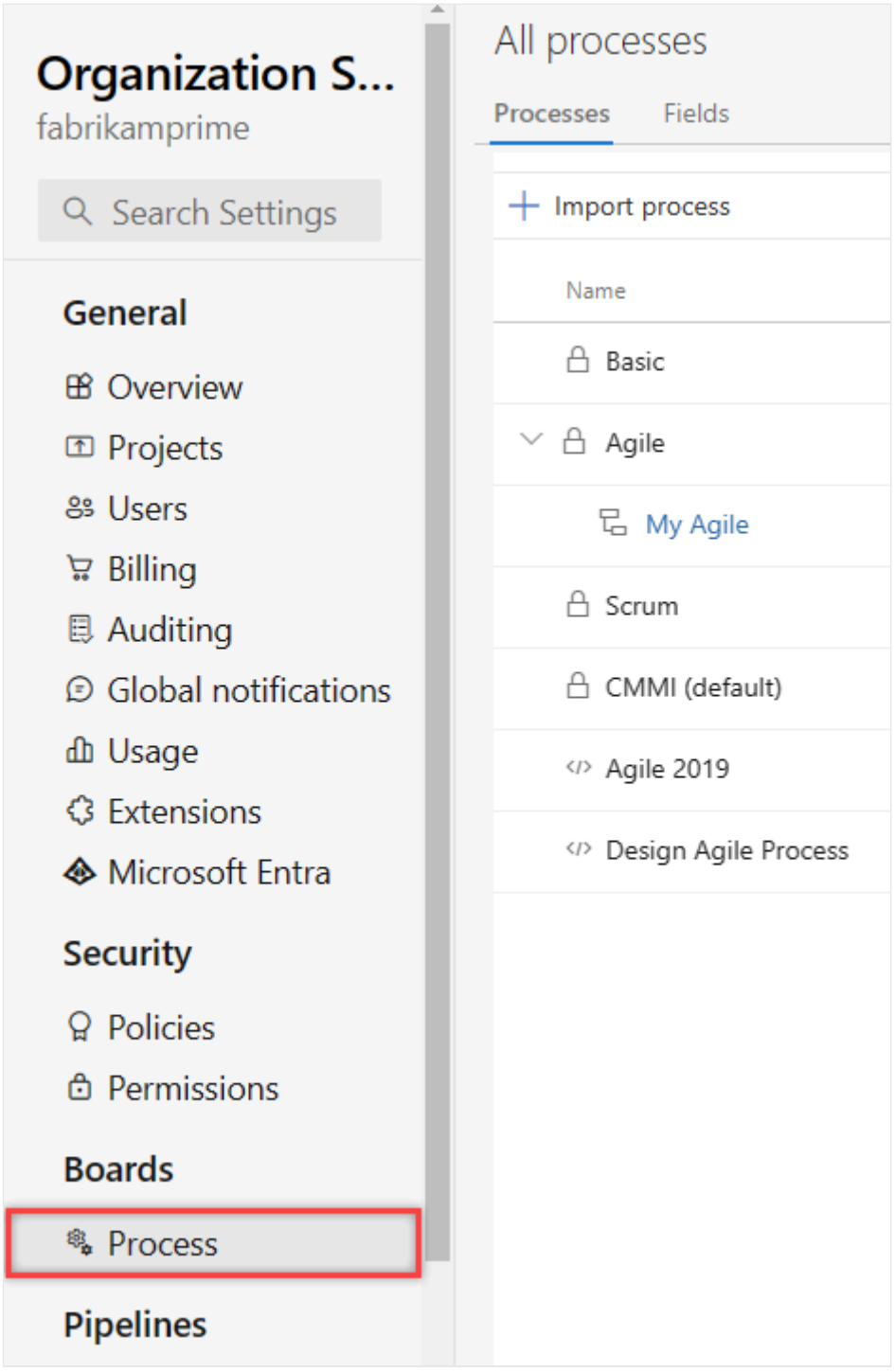
Open organization process settings

1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`).

2. Select **Organization settings**.



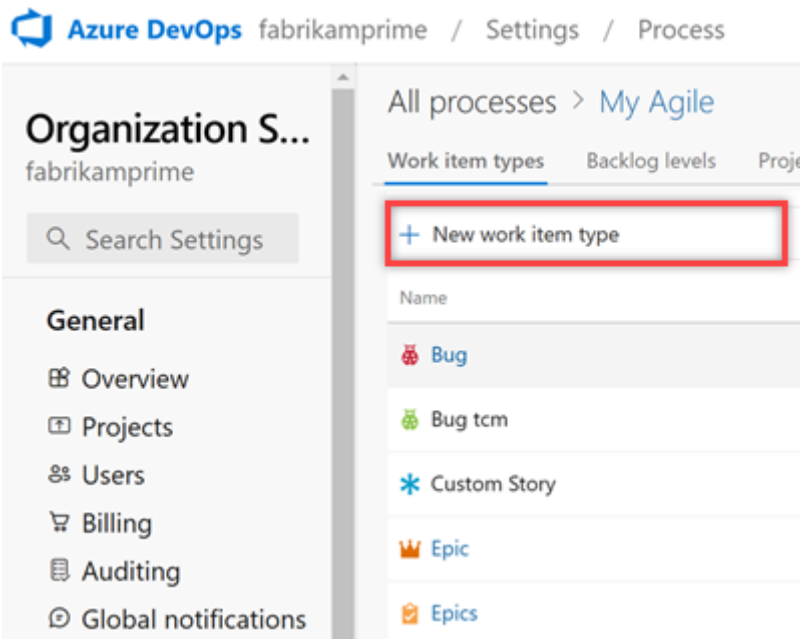
3. Select **Process**.



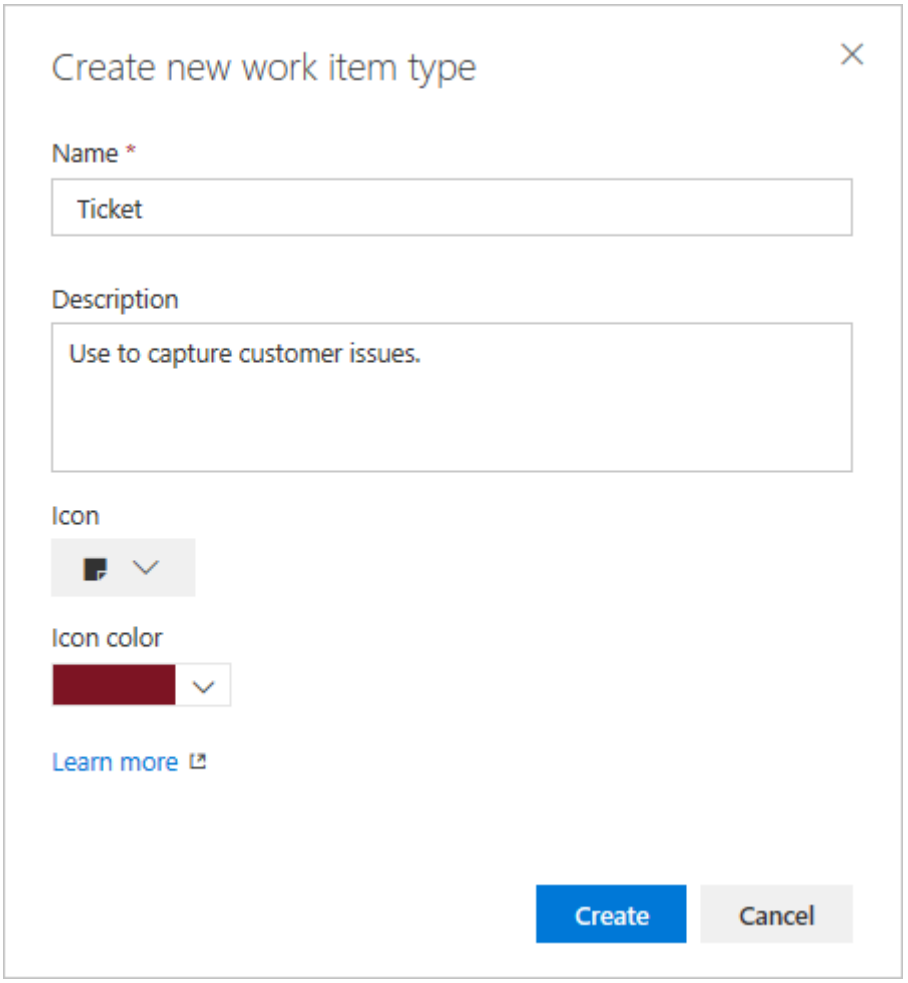
> **Note**
>
> When you customize an inherited process, any projects using that process automatically reflect the customizations. To ensure a smooth transition, we recommend creating a test process and project, which allows you to test your customizations before you implement them organization-wide. For more information, see Create and manage inherited processes.

Add a work item type

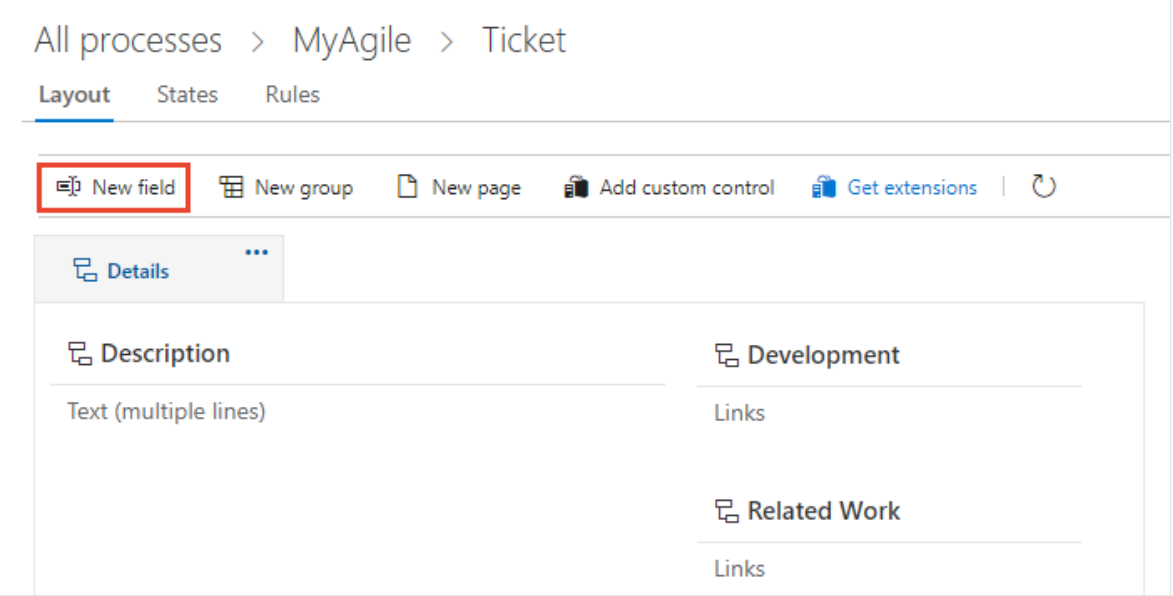
1. From the **Work Item Types** page, choose + **New work item type**.



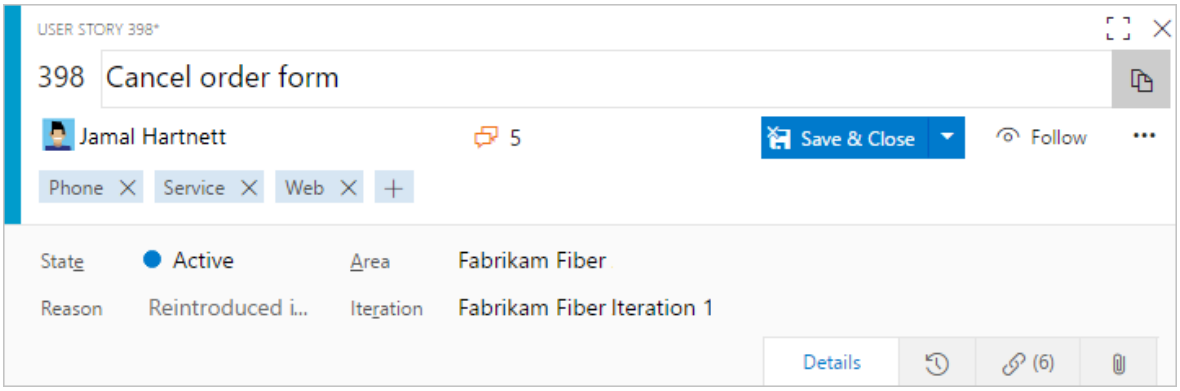
2. Name the WIT and optionally specify a description, icon and color. The icon and color you specify appear throughout the web portal, including on the work item form and when associated work items appear on a backlog, boards, query results, and more. Choose **Create** to save.



3. From the Layout page, you can add fields, groups, or pages to specify the form and fields you want to track. Each new WIT comes predefined with a Details page with the Description field, and Discussion, Development, and Related Work groups. For details, see Add and manage fields or Customize the web layout for a process.



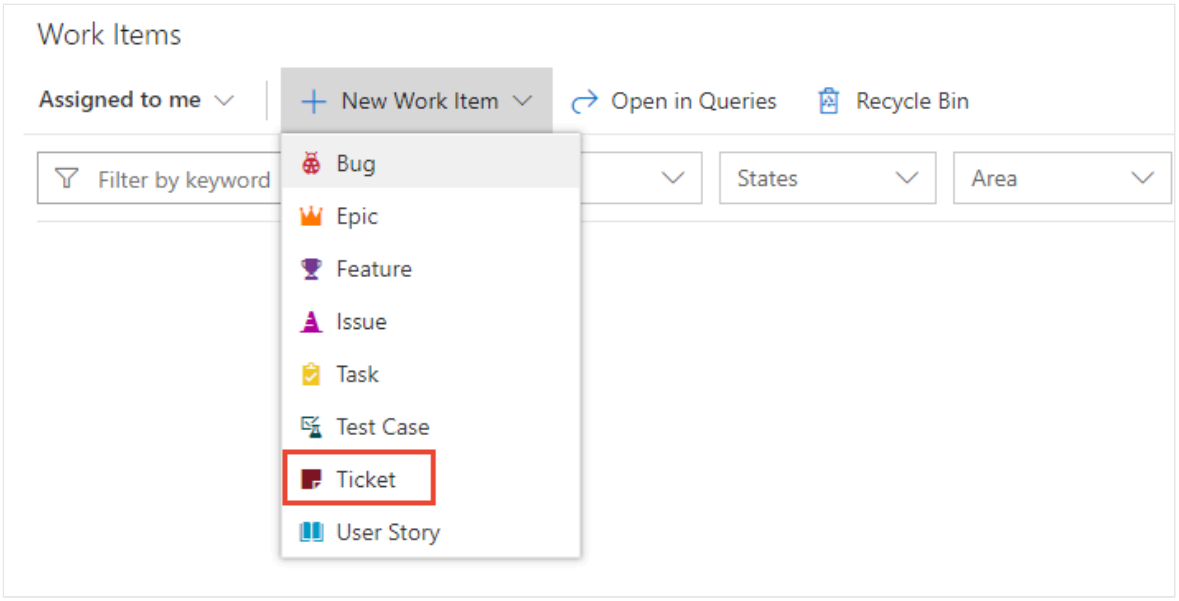
Also added, but not shown nor editable from the **Layout** page, are the standard elements included with the header of the form as shown in the following image, as well as the history, links, and attachment pages.



4. Open the States page to view the default workflow states. Optionally, customize the workflow states.

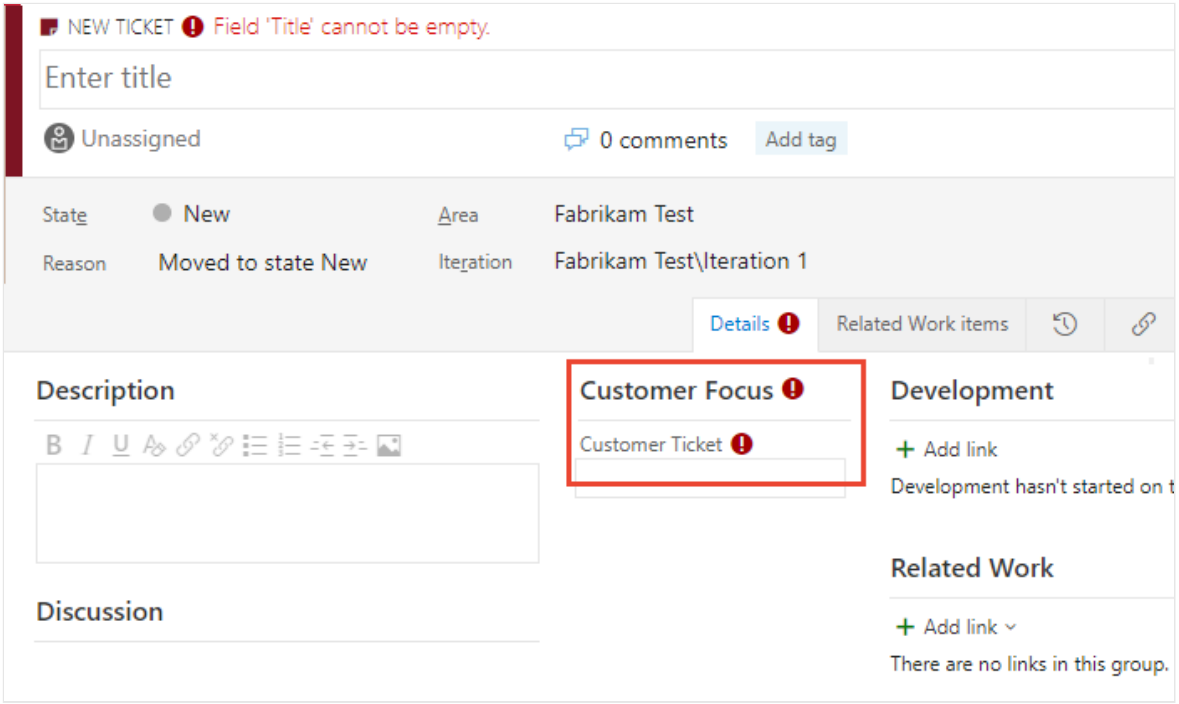
5. Optional. To add the WIT to a backlog, see Customize your backlogs or boards for a process. By default, custom WITs aren't added to any backlog. To learn more about different backlog levels, see Backlogs, boards, and plans.

6. Verify the custom work item type appears as expected. Open the **Work>Work Items** page (user context) and choose **New Work Item** and select the WIT you customized. Here we choose **Ticket**.



If you don't see the custom WIT, refresh your browser to make sure it registers all the custom changes you've made.

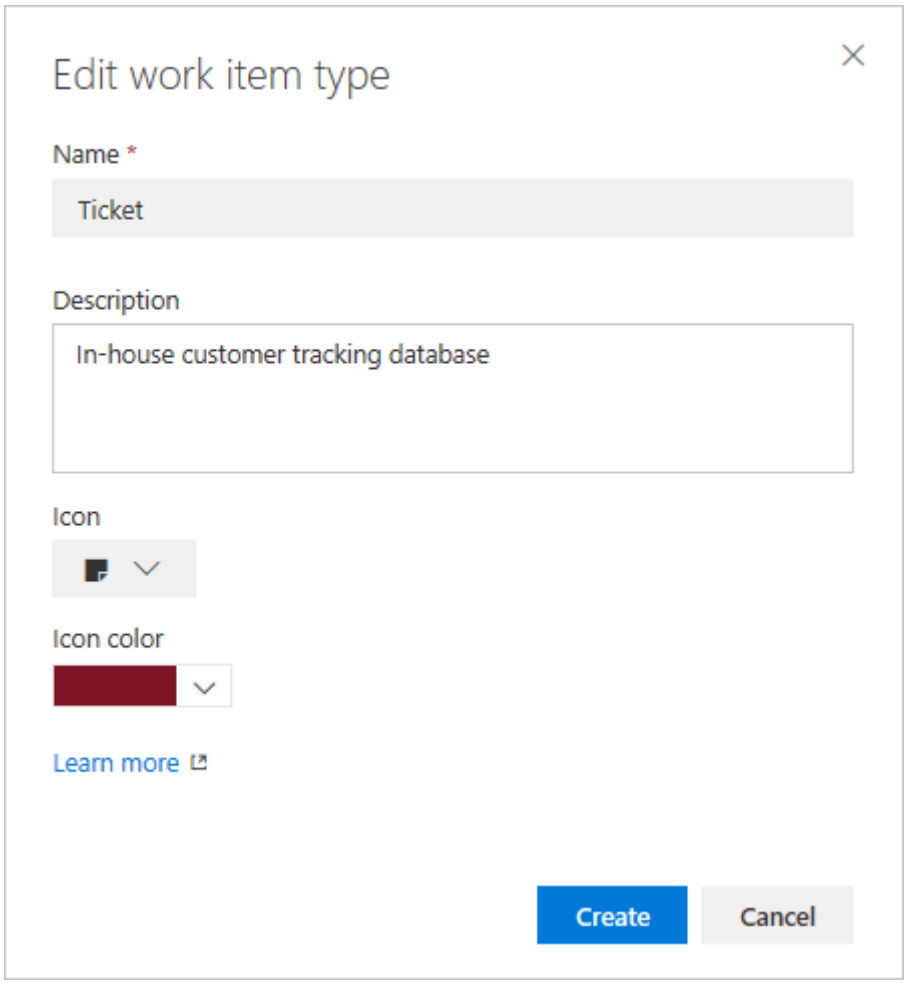
7. Verify that the field you added appears on the form. Note that the ❗ (exclamation mark) icon indicates the field is required.



Change description, icon or color

To change the description, icon or color of a custom WIT, choose Edit from the WIT context menu.

Here we change the description, icon and color for the Ticket custom WIT.



Enable or disable a WIT

You can disable a custom WIT, which prevents users from being able to add work items using it. However, all existing work items defined using the custom WIT remain available. You can query for them and edit them.

You may want to disable a custom WIT until you have defined all the fields, workflow, and layout that you have planned.

To disable a custom WIT, choose **Disable** from the WIT context menu.

To re-enable the WIT, choose **Enable** from the WIT context menu.

> ⓘ **Note**
>
> Disabling a WIT removes the WIT from the **New** drop down and add experiences. Creating a work item of that WIT is also blocked via REST APIs.
>
> No changes are made to existing work items of that type. You can update or delete them and they continue to appear on backlogs and boards. Both work item types need to be ebabled in order to do a change type operation.

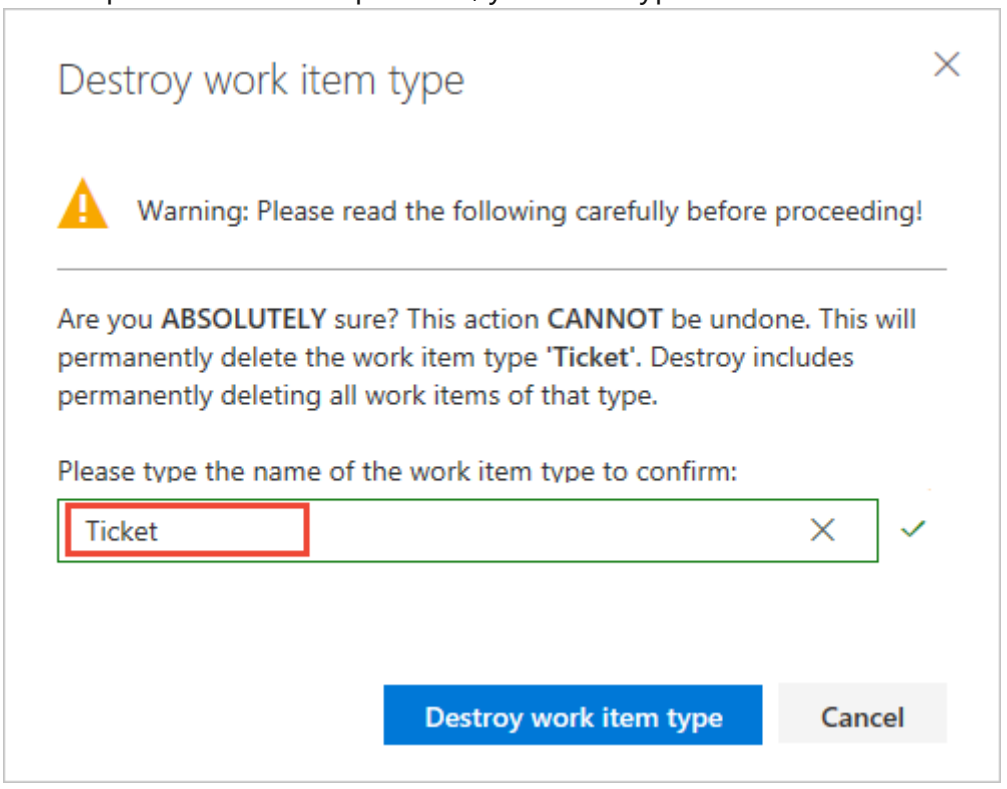
Delete or destroy a custom WIT

> **Important**
>
> Destroying a WIT deletes all work items and data associated with that WIT, including historical values. Once destroyed, you can't recover the data.

1. To completely remove a custom WIT and all work items based on that WIT, choose **Destroy** from the WIT context menu.

   To destroy a WIT, you must be a member of the Project Collection Administrators group or be granted explicit permissions to edit a specific process.

2. To complete the delete operation, you must type the name of the WIT as shown.



WIT extensibility

To work with WITs programmatically, see Work Item Types REST API.

Q & A

Q: How do I get my custom work item type to show up on my backlog?

A: Modify your requirement backlog to include the custom work item type. For details, see Customize your backlogs or boards, Edit or rename the requirement backlog.

> **Note**
>
> The backlog level to which you add a custom work item type determines the parent work item type(s) for the work item type.

Related articles

> **Note**
>
> Review changes made to an inherited process through the audit log. For more information, see Access, export, and filter audit logs.

- Add and manage fields
- Customize the layout
- Customize a workflow for a work item type
- Customize a project using an inherited process

Feedback

Was this page helpful? Yes No

Provide product feedback

Customize backlogs and boards (Inheritance process)

Article • 07/17/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

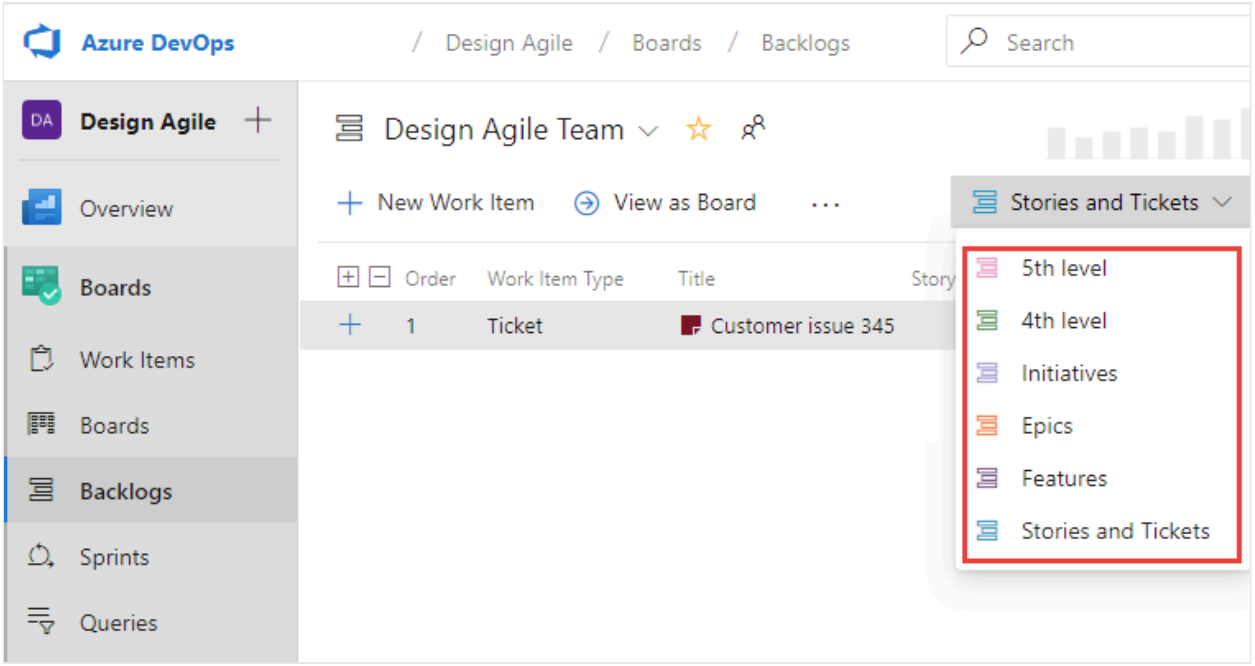
In your project, you currently have two predefined portfolio backlogs: "Features" and "Epics." But, if your project requires more portfolio backlogs, you can create them.

Benefits of portfolio backlogs:

- **Organizing work:** Portfolio backlogs allow you to organize work based on business initiatives, user scenarios, or other relevant criteria.
- **Hierarchical view:** By structuring backlogs into portfolios, you gain a hierarchical view of work, which includes items defined in lower-level backlogs (such as user stories, features, or tasks).
- **Cross-team visibility:** Program managers can track the status of backlog items across multiple teams. They can drill down to ensure that all work is adequately represented.

For more information, see About process customization and inherited processes.

In the following example, we added a third level portfolio backlog labeled *Initiatives*, which tracks the custom *Initiative* work item type. We also renamed the product backlog to *Stories and Tickets* to indicate that we not only track *User Stories*, but also *Customer Tickets* on the product backlog.



Supported customizations

Backlogs and boards are essential Agile tools for creating and managing work for a team. The standard backlogs—product, iteration, and portfolio—inherited from the system process are fully customizable. In addition, you can add custom portfolio backlogs for a total of five portfolio backlogs.

Backlog types

Customization support

Inherited backlogs

- Add a custom work item type (WIT)
- Add an inherited work item type
- Change the default work item type
- Rename a backlog

Custom portfolio backlogs

- Add a portfolio backlog that displays custom work item types (WITs)
- Edit or rename a portfolio backlog
- Delete the top-level custom portfolio backlog

Unsupported customizations:

- **Removing an inherited portfolio level:**
  - While you can't directly remove an inherited portfolio level from a product, you have a couple of options:
    - **Rename the portfolio level:** You can rename the inherited portfolio level to better suit your needs.
    - **Disable an inherited WIT:** If the inherited portfolio level includes WITs that you don't want to use, you can disable them. This action prevents teams from creating new work items of those types.
- **Inserting a backlog level:**
  - You can't insert a new backlog level within the existing set of defined backlogs. The predefined backlog levels are typically fixed (for example, Epics, Features, User Stories, Tasks), and you can't add custom ones in between.
- **Reordering backlog levels:**
  - Unfortunately, you can't reorder the backlog levels. They usually follow a predefined hierarchy, and changing their order isn't supported.
- **Adding a WIT to multiple backlog levels:**

    - Each WIT can only belong to one backlog level. You can't add a WIT to two different backlog levels simultaneously.
- **Creating a custom task backlog level:**
    - Although you can't create a custom task-specific backlog level, you can still add custom WITs to the iteration backlog. For example, you could create a custom WIT called "Enhancement" or "Maintenance" and associate it with the iteration backlog.
- **Managing bugs:**
    - The Bug WIT doesn’t belong to any specific backlog level by default. Instead, each team can decide how they want to manage bugs. You can choose to show bugs on backlogs and boards or handle them separately.

Add a system work item type to a backlog

If you want to track Issues or Impediments or other inherited work item types within a backlog or board, edit the corresponding backlog. The following table lists the available work item types you can add to a backlog.

Process	Work item types
Agile	Issue
Scrum	Impediment
CMMI	Change Request, Issue, Review, Risk

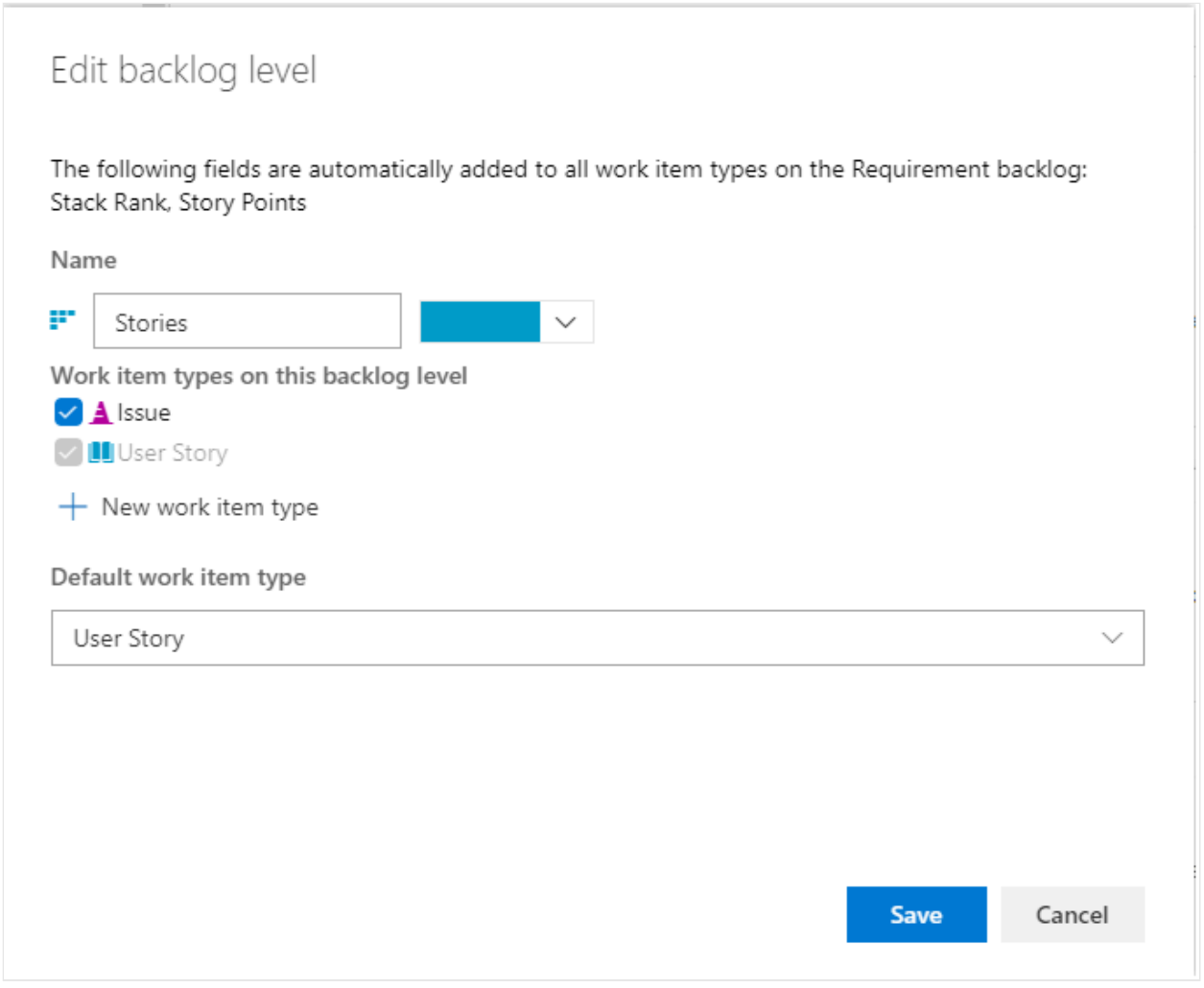
Each **Edit backlog level** dialog automatically includes inherited and custom work item types that aren't assigned to other backlog levels. For example, unassigned Agile work item types are listed under the **Other work item types** section as shown in the following image

Other work item types

Bug work items appear on the requirement backlog or the iteration backlog based on the team's setting. The remaining work item types are not displayed on any backlog or board.

Backlog	Work item types
Requirement or Iteration backlog	Bug
No associated backlog	Issue

These same work item types, along with any custom work item types, appear in the **Edit backlog level** dialog of all backlog levels, until they get assigned to a particular backlog level.



> **Note**
>
> You can't remove the default, inherited work item type from any backlog level, but you can disable the corresponding work item type. For example, you can disable the User Story work item type for the Agile Requirement backlog as long as you have added another work item type to support that backlog.

Fields added to work item types

When you add a WIT to a backlog level, certain fields are automatically added to the WIT definition as hidden fields. These fields don't appear on the work item form but are essential for supporting specific Agile tool features.

Expand table

Backlog level	Fields added	Description
Portfolio backlog	- Stack rank (Agile, CMMI) - Backlog Priority (Scrum)	The Stack Rank and Backlog Priority fields capture the relative priority of work items as they get reordered on a backlog or board. For more information, see Behind the scenes: the Backlog Priority or Stack Rank field.
Requirement backlog	- Stack Rank, Story Points (Agile) - Stack Rank, Size (CMMI) - Backlog Priority, Effort (Scrum)	The Story Points, Size, and Effort fields capture the relative work required to complete a WIT assigned to the Requirement backlog. This value is used to compute velocity.
Iteration backlog	- Activity, Remaining Work, Stack Rank (Agile) - Discipline, Remaining Work, Stack Rank (CMMI) - Activity, Remaining Work, Backlog Priority (Scrum)	Remaining Work is used in Sprint burndown and capacity charts.

Prerequisites

We recommend checking out Configure and customize Azure Boards, which offers guidance on tailoring Azure Boards to align with your specific business requirements.

- You must have an organization in Azure DevOps.
- To create, edit, and manage processes, you must be a member of the **Project Collection Administrators** group or have the corresponding collection-level permissions, **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization**, set to **Allow**. For more information, see Set permissions and access for work tracking, Customize an inherited process.

> **Note**
>
> When you customize an inherited process, any projects using that process automatically reflect the customizations. To ensure a smooth transition, we recommend creating a test process and project, which allows you to test your customizations before you implement them organization-wide. For more information, see Create and manage inherited processes.

Add or edit portfolio backlogs

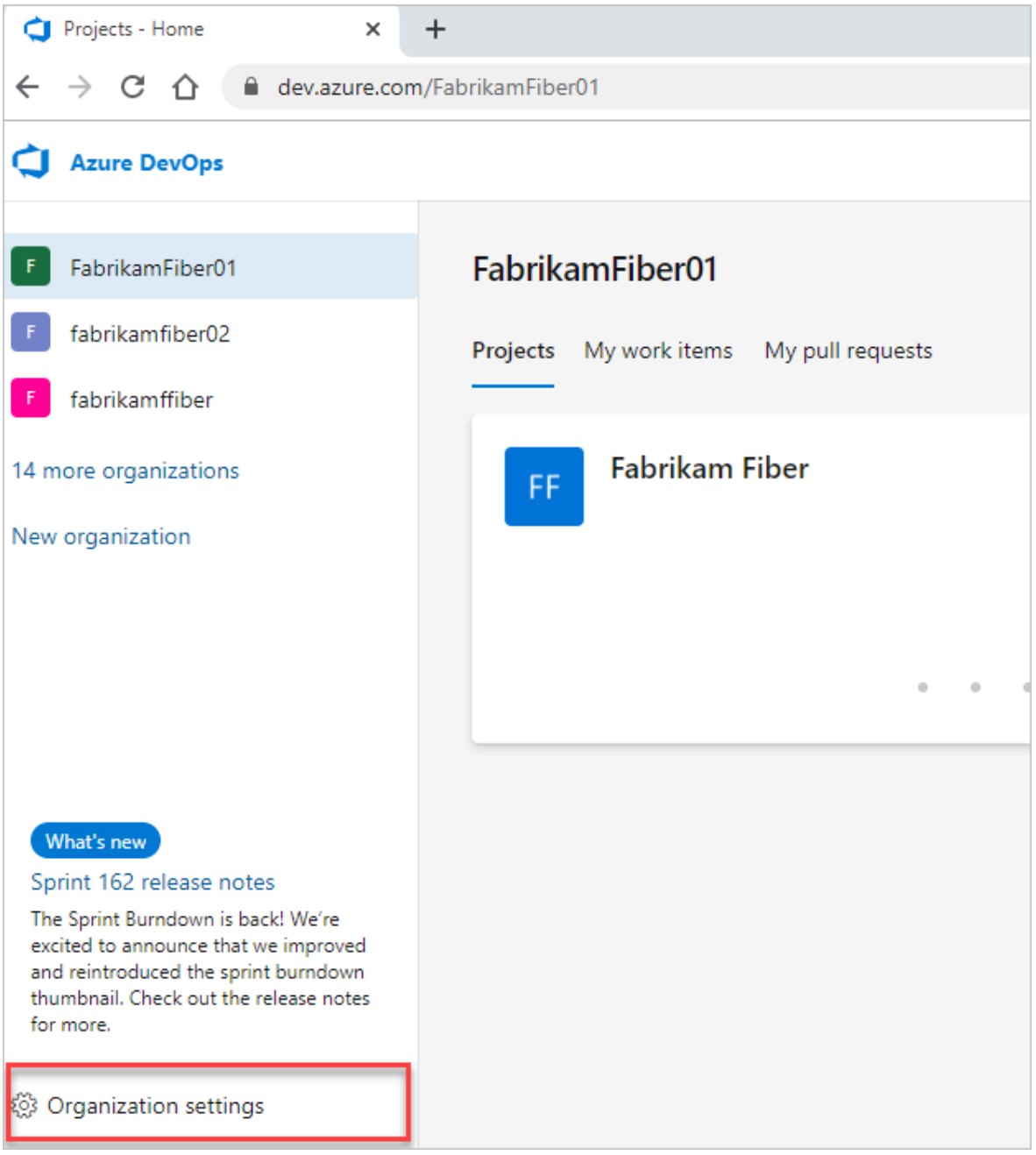
The Agile, Scrum, and CMMI system processes define two default portfolio backlogs, Epics and Features. Each is associated with their corresponding work item types, Epic, and Feature. The Basic process only defines the Epics backlog and Epic work item type. For more information, see About processes and process templates.

You can add a custom WIT or select one that you previously added. Keep in mind that only WITs not associated with another backlog level appear for selection.

Add a portfolio backlog

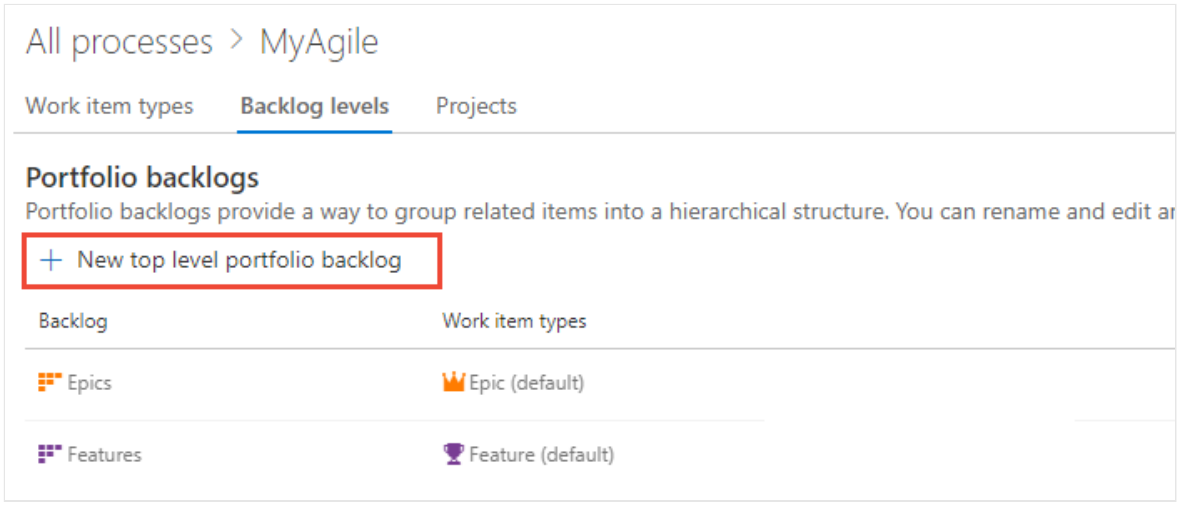
You can add a portfolio backlog and custom work item type following these steps.

1. Sign in to your organization (`https://dev.azure.com/{Your_Organization}`).
2. Select **Organization settings**.

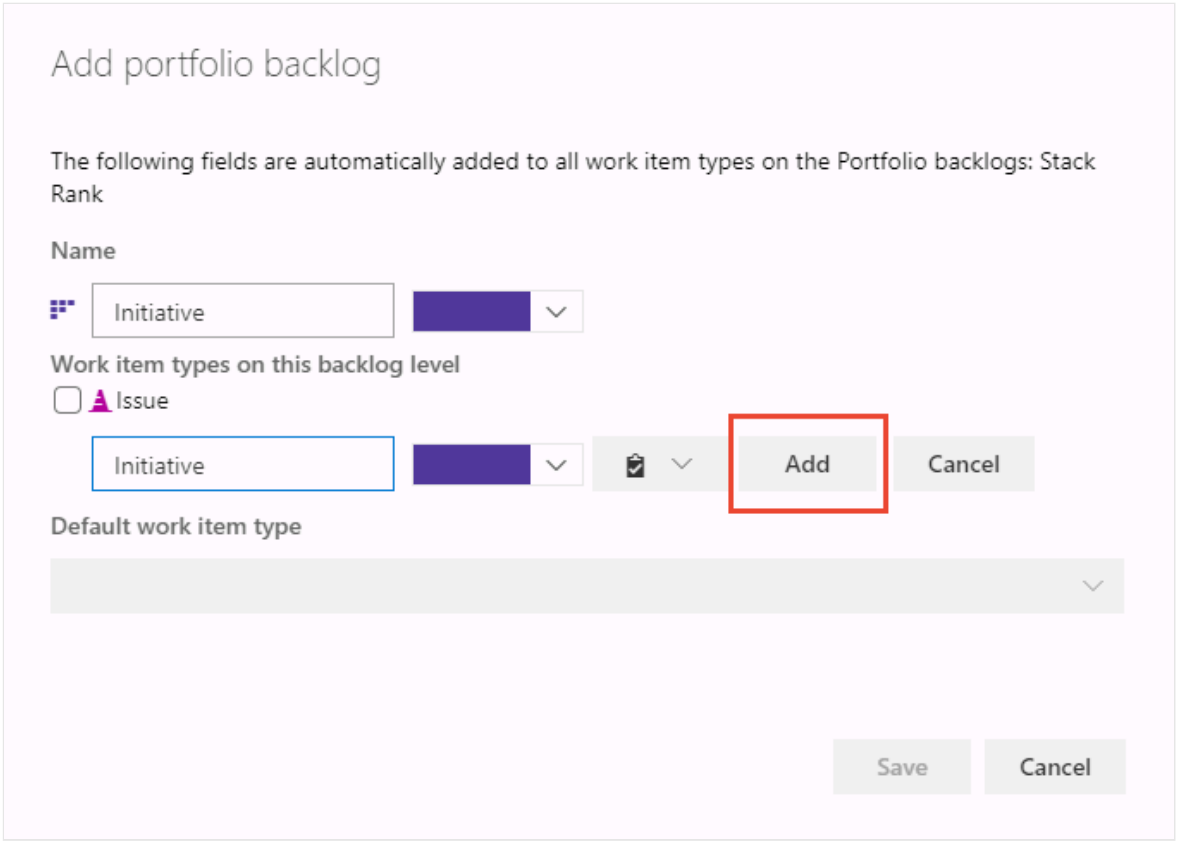


3. Select **Process**.

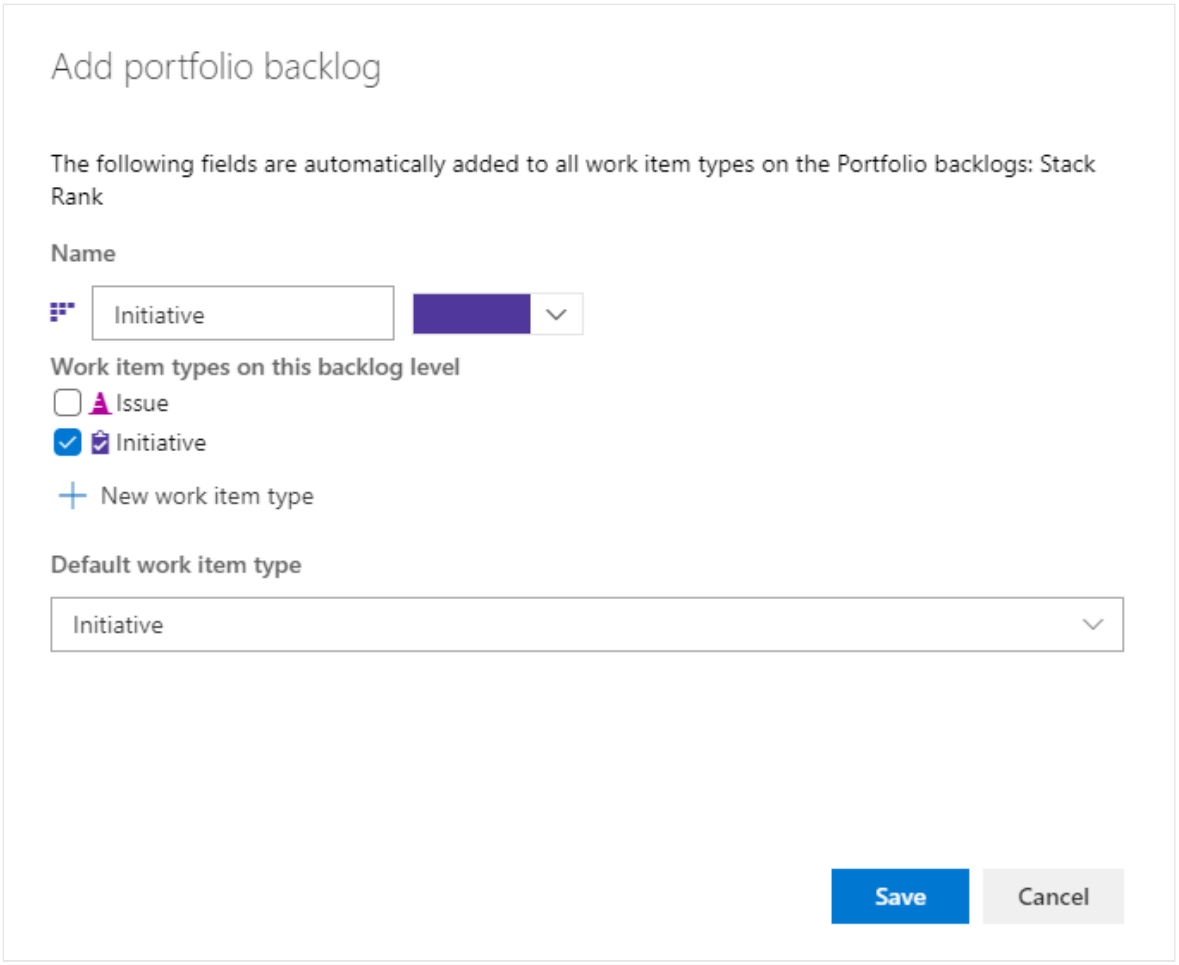
4. From the **Backlog levels** page, choose + **New top level portfolio backlog**.



5. Name the backlog level, select the backlog level color, and add the work item type to associate with this level, and then select **Add**.

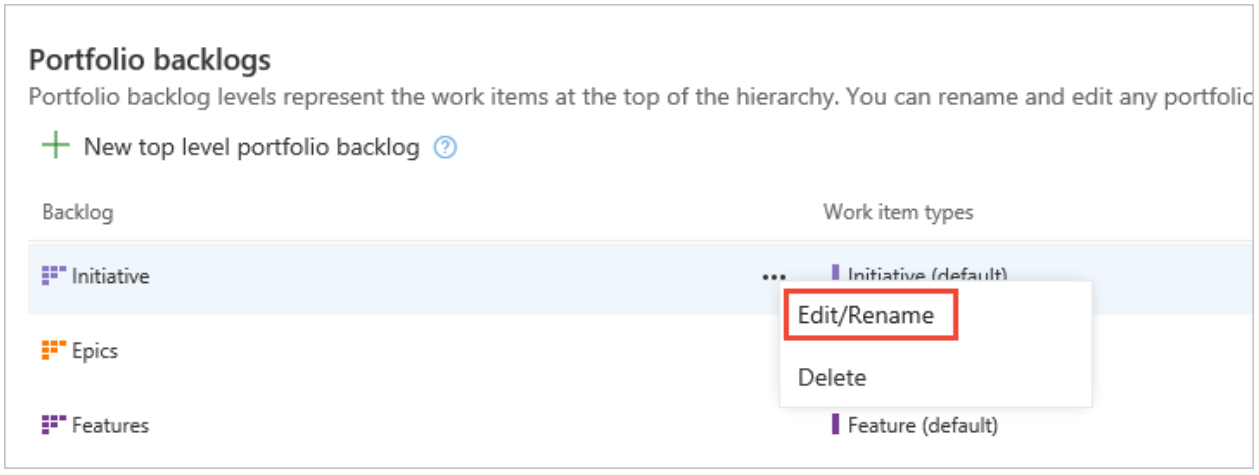


6. If you're associating only one work item type with the backlog, then choose **Save** to save your changes. Otherwise, you can add more work item types as needed.

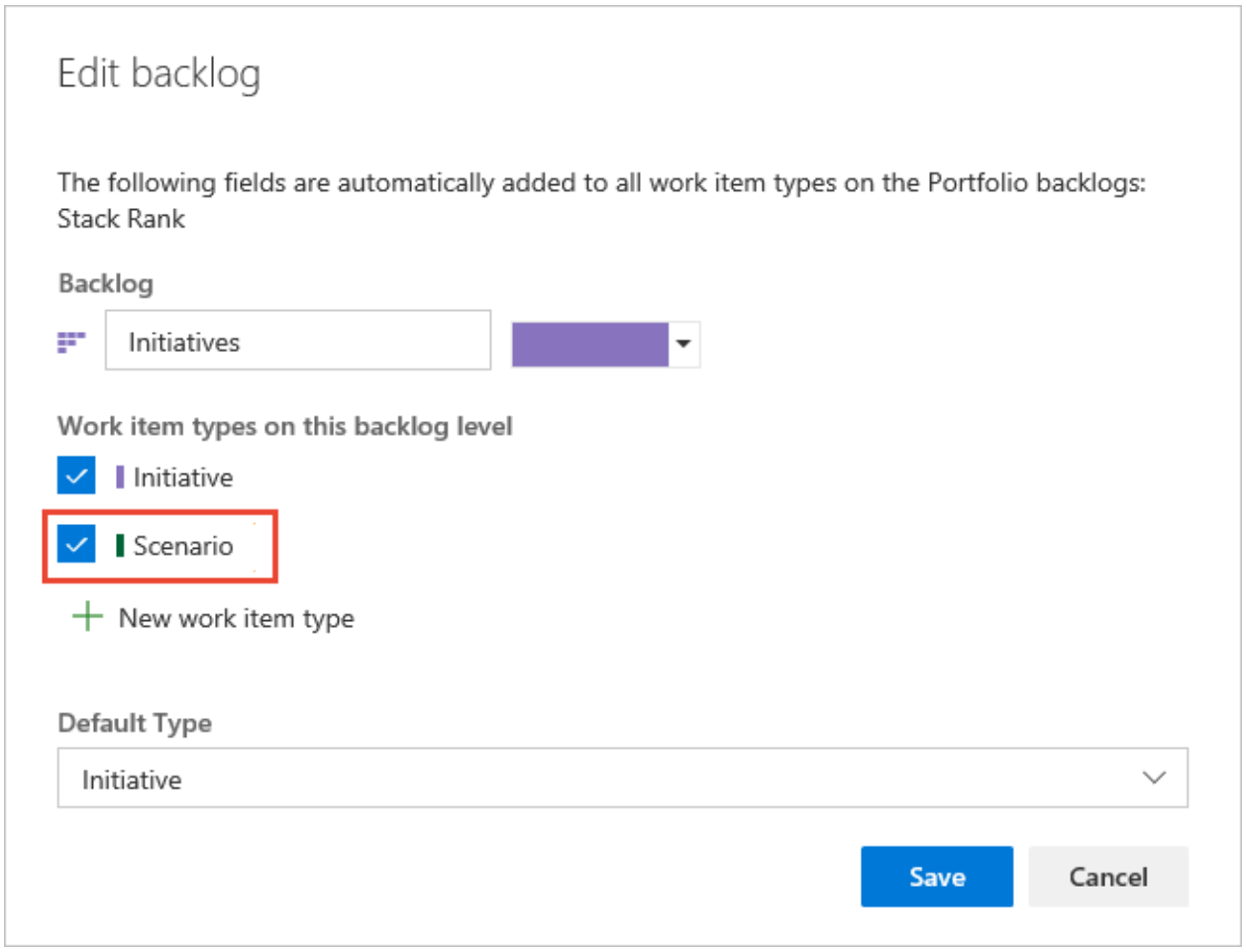


Edit, rename, or delete a portfolio backlog

From the **Backlog levels** page, choose the ··· context menu of a portfolio backlog to edit, rename, or delete it.



Deleting a backlog level removes the backlog and board associated with the level for all teams, including customizations made to them. The work items defined with the associated work item types aren't deleted or affected in any way.



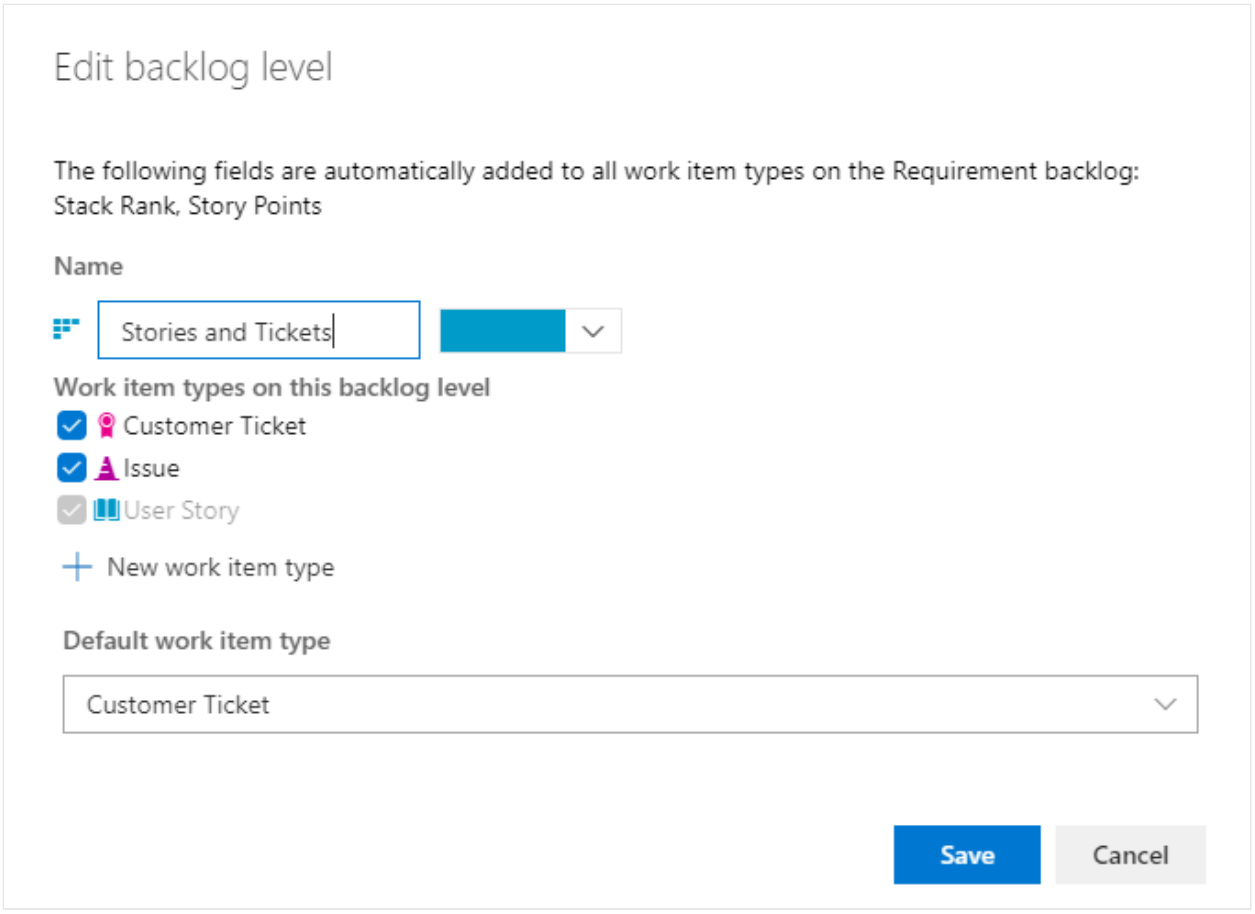
> **Note**
>
> You can't remove the default, inherited work item type from the Epics or Features portfolio backlogs. However, you can disable these work item types and that effectively removes them from the user interface.

Edit or rename the requirement backlog

The Requirement backlog, also referred to as the product backlog, defines the work item types that appear on the product backlog and board. The default work item type for Agile is User Story; for Basic, Issue; for Scrum, Product Backlog Item; and for CMMI, Requirement.

You can rename the backlog, change the color, add work item types, and change the default work item type. Open the Edit backlog dialog from the context menu for the Requirements backlog.

In the following example, we renamed the backlog, added **Customer Ticket** and **Issue**, and changed the default type to *Customer Ticket*. Check those boxes of the work item types to include on the backlog.



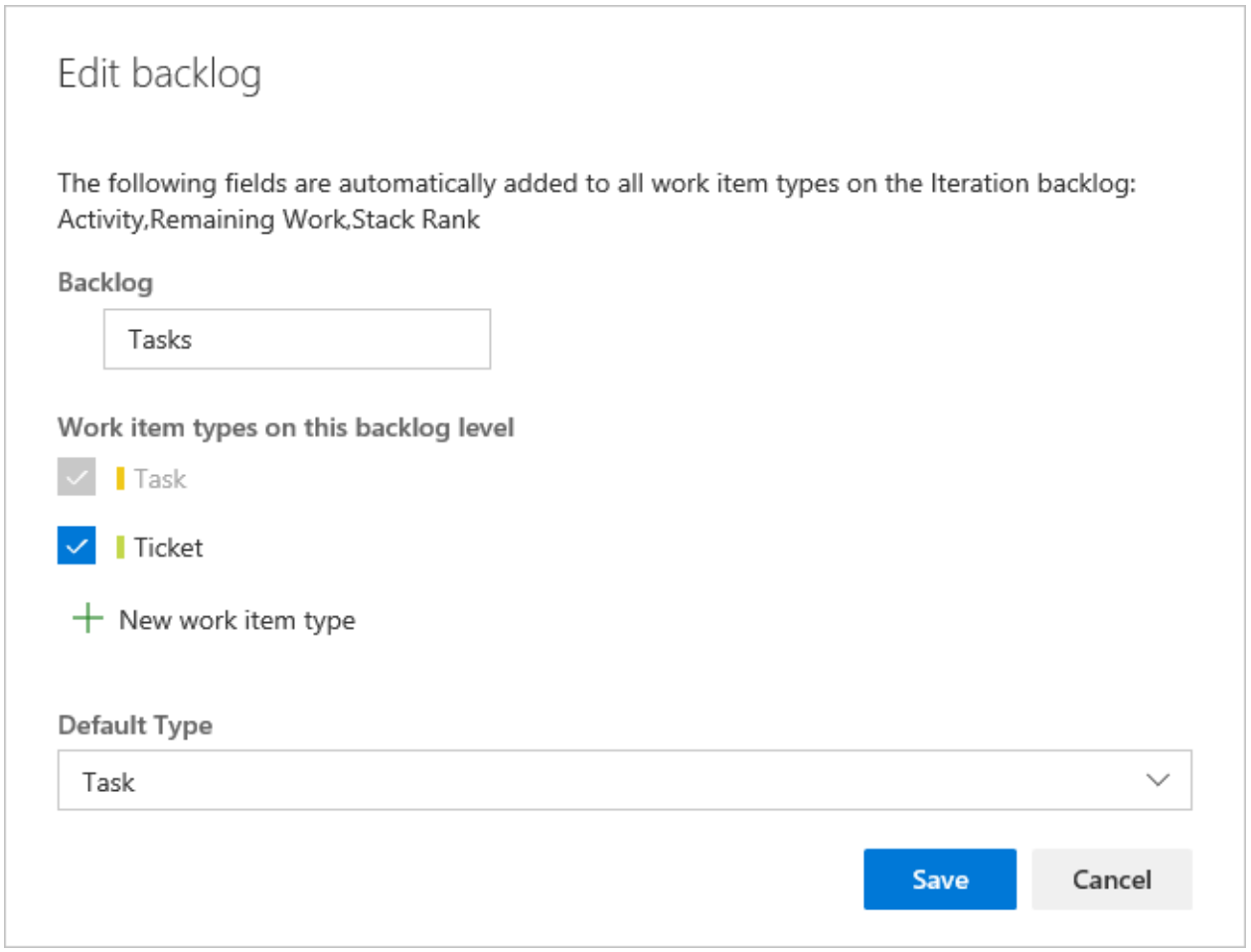
> **Note**
>
> You can't remove the default, inherited work item type from the Requirements backlog. However, you can disable the work item type and that effectively removes it from the user interface.

Edit the iteration backlog

The Iteration backlog, also referred to as the sprint backlogs, defines the work item types that are displayed on the sprint backlogs and Taskboards. The default work item type for all processes is Task.

For the iteration backlog, you can add work item types and change the default work item type. Open the Edit backlog dialog from the context menu for the Iteration backlog.

In the following example, we added the *Ticket* work item type that is tracked along with tasks.



> **Note**
>
> You can't remove the default, inherited work item type from the Iteration backlog. However, you can disable the work item type and that effectively removes it from the user interface.

Related articles

- Organize your backlog, work with portfolio backlogs
- Add a custom work item type
- Learn about backlogs, boards, and plans
- Customize a project using an inherited process
- View changes made to an inherited process through the audit log.

Feedback

Was this page helpful? Yes No

Provide product feedback

Customize the web layout for a work item type (Inheritance process)

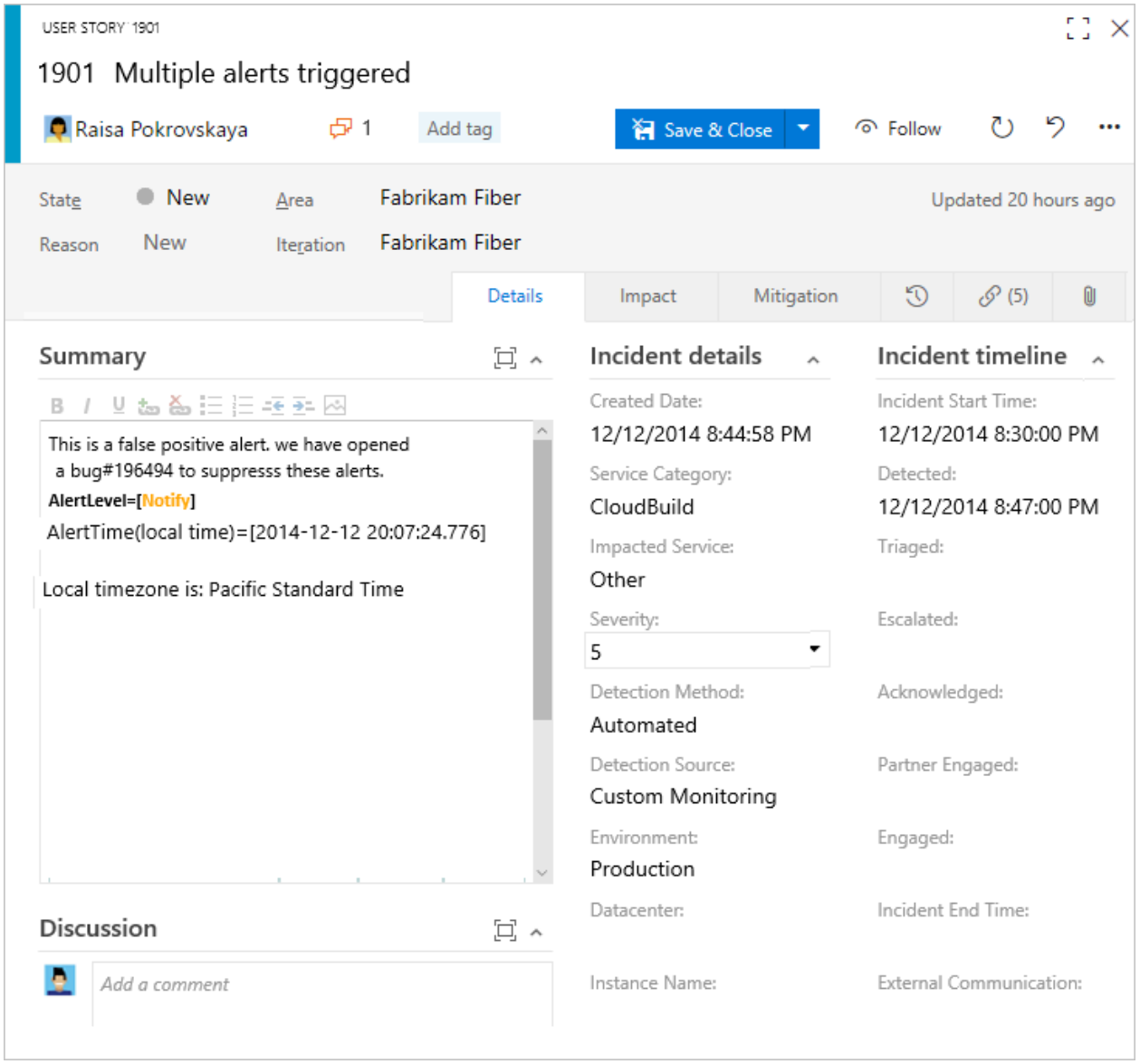
Article • 07/24/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Work items are the backbone for planning and tracking work. As such, you'll want to make sure that each work item type (WIT) is tracking the data that your team needs.

To support your work tracking requirements, you can customize each WIT to add custom groups to collect custom fields that you add under a common section. You can organize your forms by placing logically related groups and HTML fields on separate pages within a form to minimize clutter on any one page within the form.

For example, the following form shows that two custom pages have been added, **Impact** and **Mitigation**. In addition, several new groups and fields have been added to the Details page.



Once you've added a custom page or group, you can add custom fields to those pages or groups. To learn more about what you can customize, see About process customization and inherited processes.

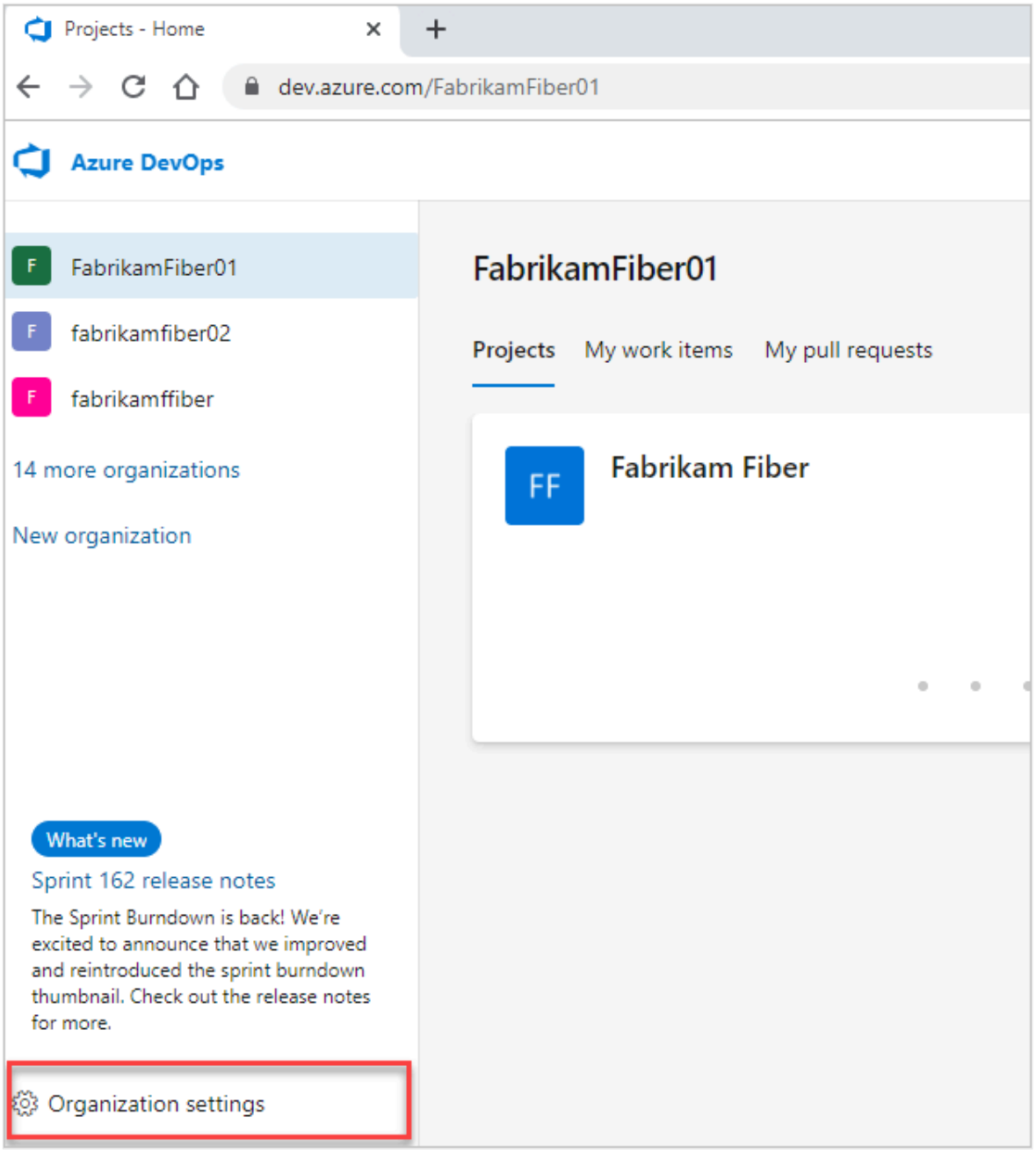
Prerequisites

Check out Configure and customize Azure Boards, which offers guidance on tailoring Azure Boards to align with your specific business requirements.

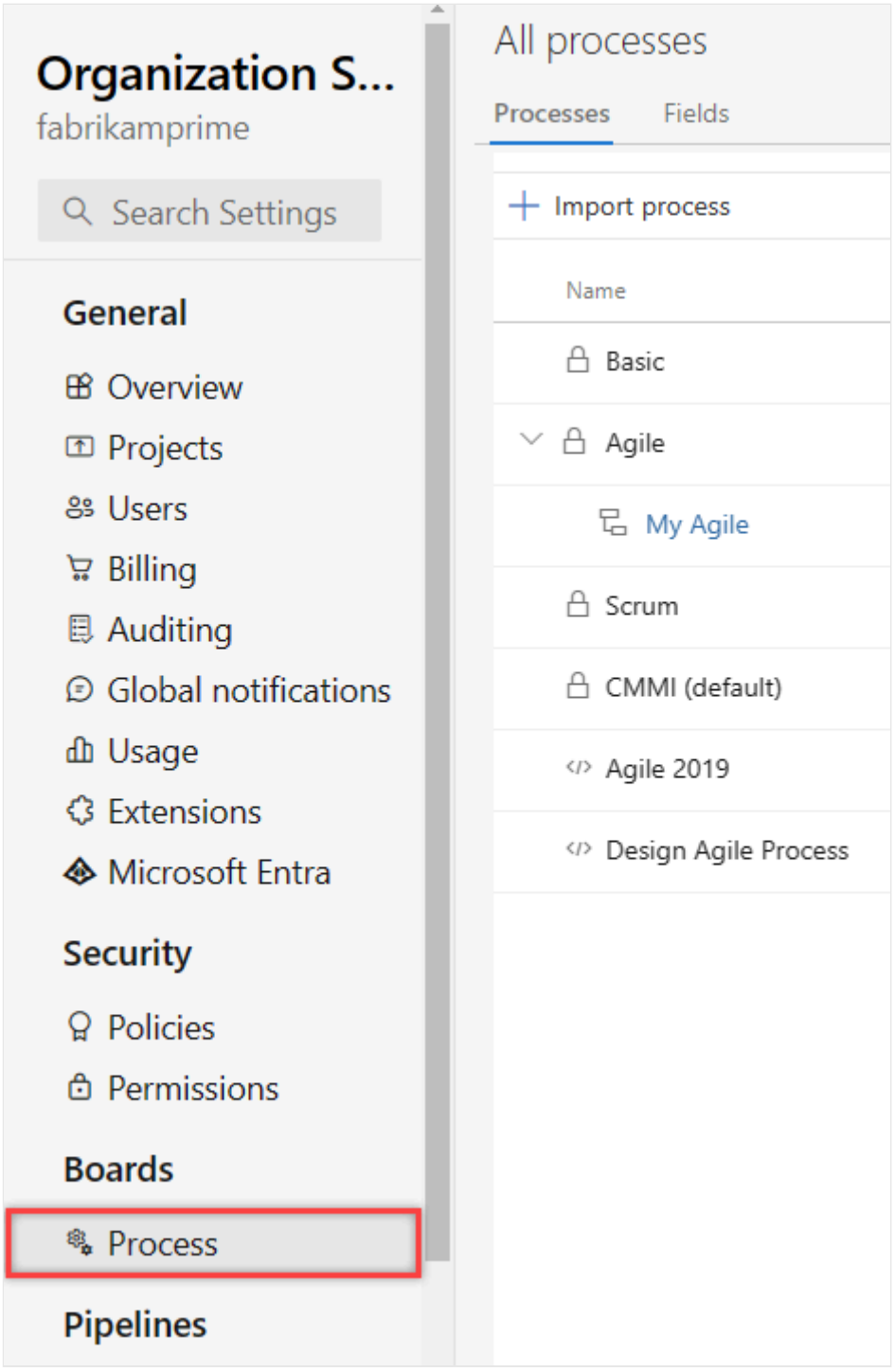
- **Organization requirement**: Ensure you have an organization in Azure DevOps.
- **Permissions**:
  - Be a member of the **Project Collection Administrators** group.
  - Have collection-level permissions such as **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**.
  - These permissions allow you to modify processes and fields within your organization.

Open organization process settings

1. Sign in to your organization (`https://dev.azure.com/{yourorganization}`).
2. Select **Organization settings**.



3. Select **Process**.

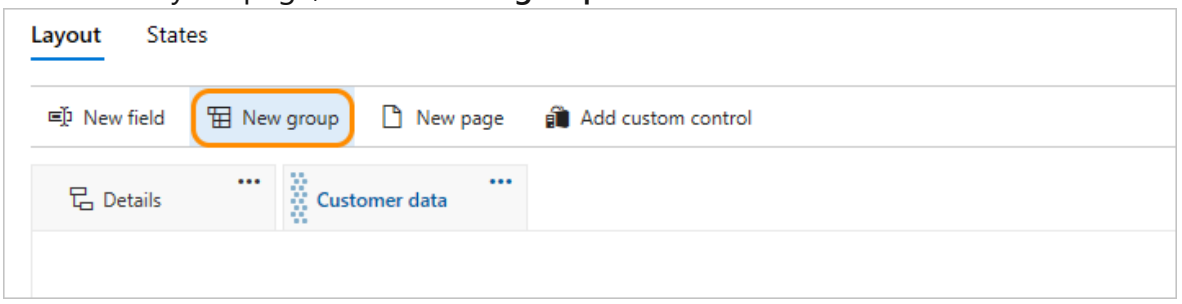


> **Note**
>
> When you customize an inherited process, any projects using that process automatically reflect the customizations. To ensure a smooth transition, we recommend creating a test process and project, which allows you to test your customizations before you implement them organization-wide. For more information, see Create and manage inherited processes.

Add a custom group

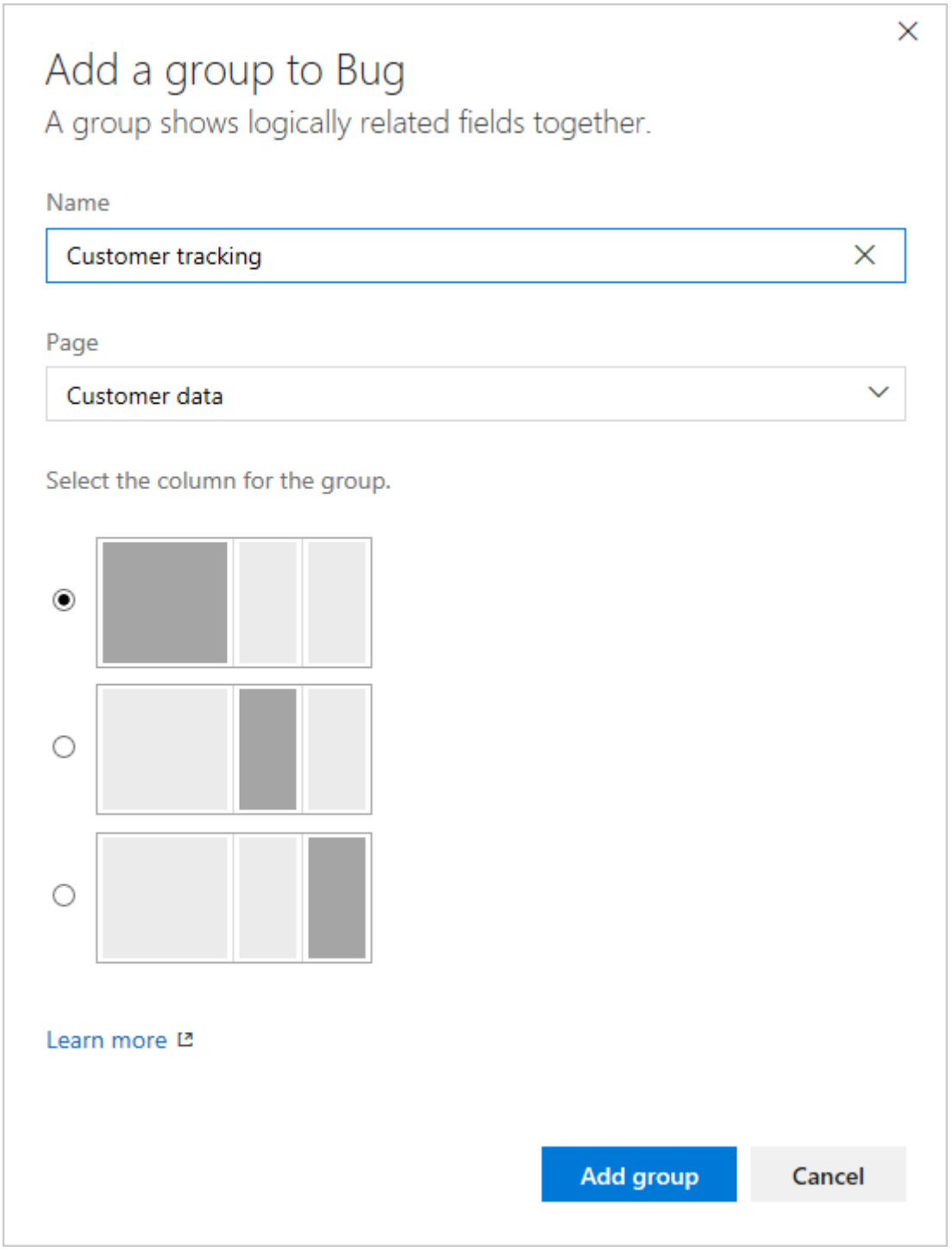
You can add your own custom groups to both inherited pages and custom pages.

1. From the Layout page, choose **New group**.



   If the **New field** and other options are disabled, you don't have the necessary permissions to edit the process. See Set permissions and access for work tracking, Customize an inherited process.

2. Name the group and choose the column it should appear in. For information about columns, see About process customization and inherited processes, Layout and resizing.



3. (Optional) Once you've added the group, you can drag and drop the group into any column or within the column, or select the Move Up or Move Down options from the group's ... context menu.

Delete or remove a custom group

Before you remove a group, first move all fields defined in that group to another group or remove them from the form.

> **Tip**
>
> You must remove all fields from a group for the Remove option to appear.

Choose the ✕ Remove option from the custom group ... context menu.

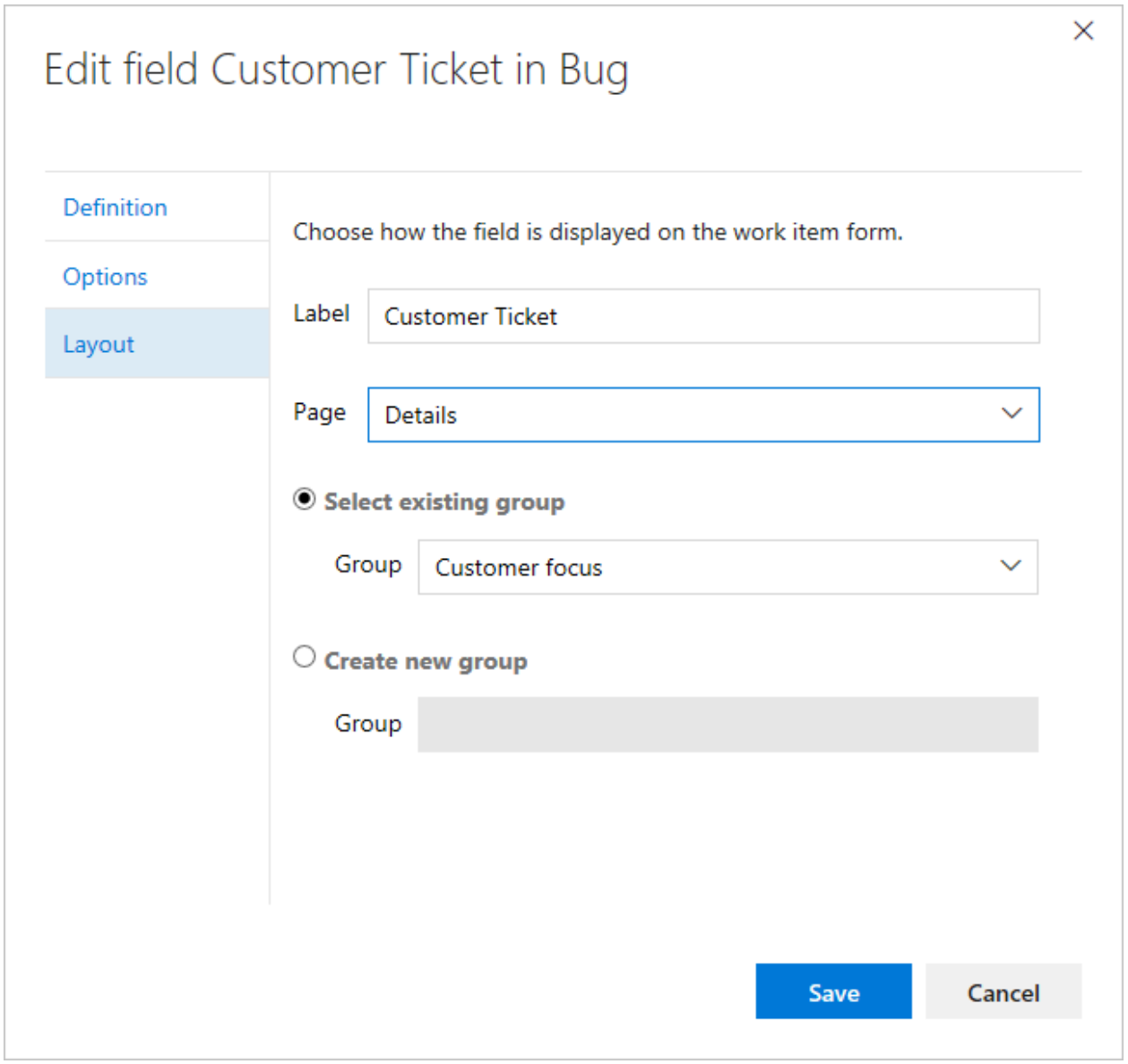
> **Note**
>
> When all fields in an inherited group are hidden, the inherited group itself does not appear on the form.

Move a field

Within a page, you can quickly drag-and-drop a field to a new group or section within the page.

To move a field to another page, choose Edit from the field's ... context menu and then choose the Page from the menu. Optionally, select an existing group or create a new group.

Here we move the Customer ticket field to the Details page and the Customer focus group.



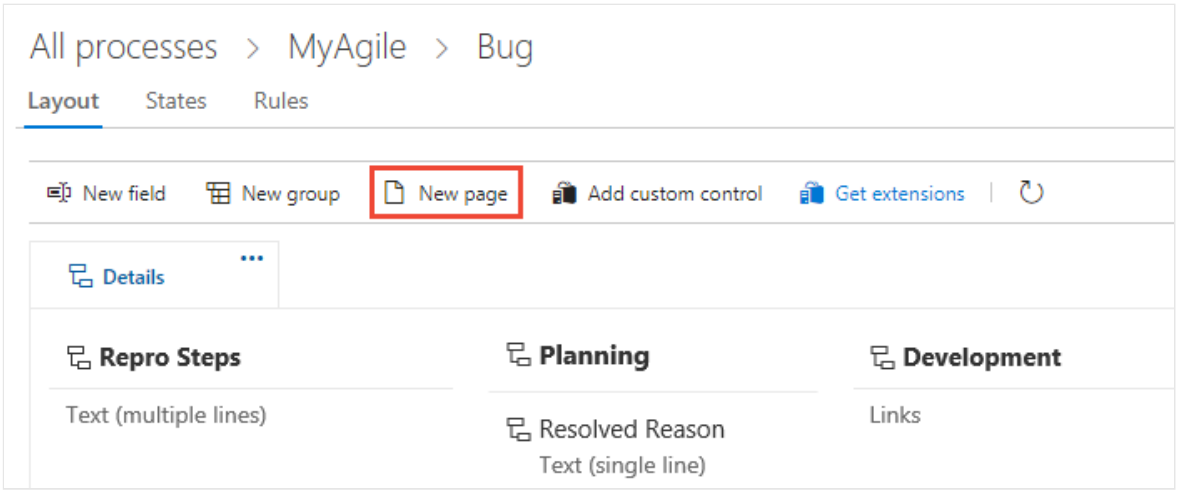
Remove a field from a form

You can only remove custom fields from a form. For inherited fields, you can hide them from showing on the form. For more information, see Customize a field for a process, Show, hide, or remove a field.

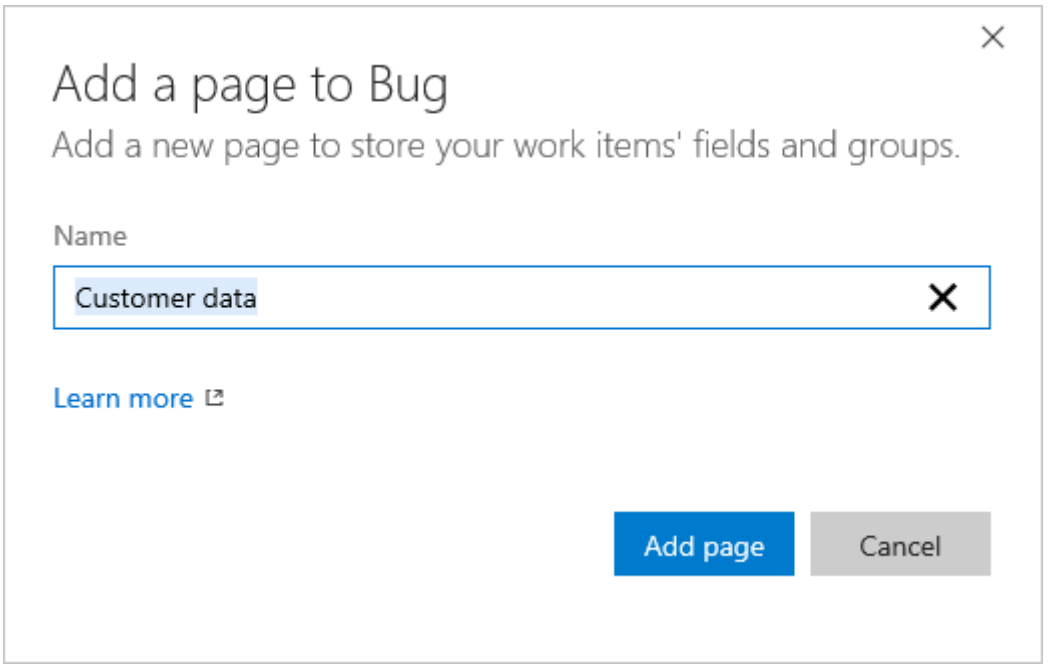
Add a custom page

Add a custom page when you have several custom fields that you want to group within that page. You can add and reorder custom pages.

1. From the Layout page of the WIT you've selected, click **New page**.

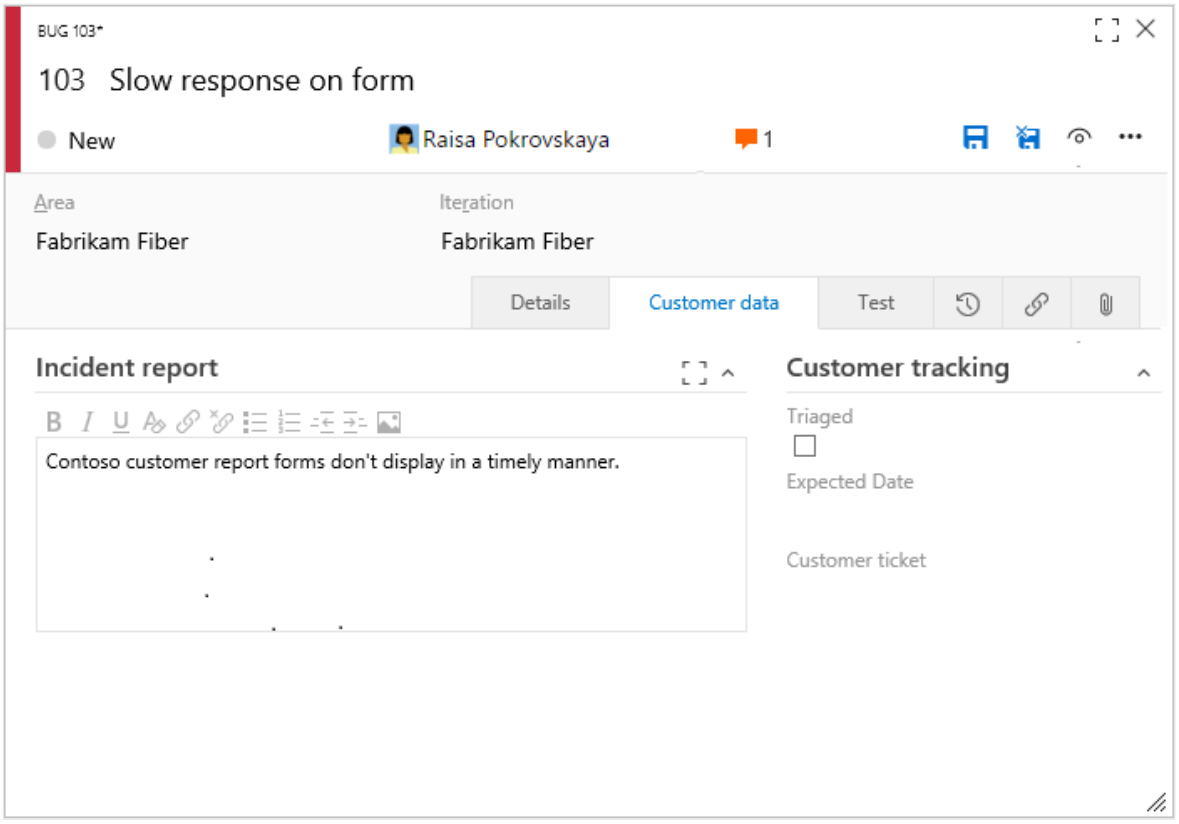


2. Label the page.



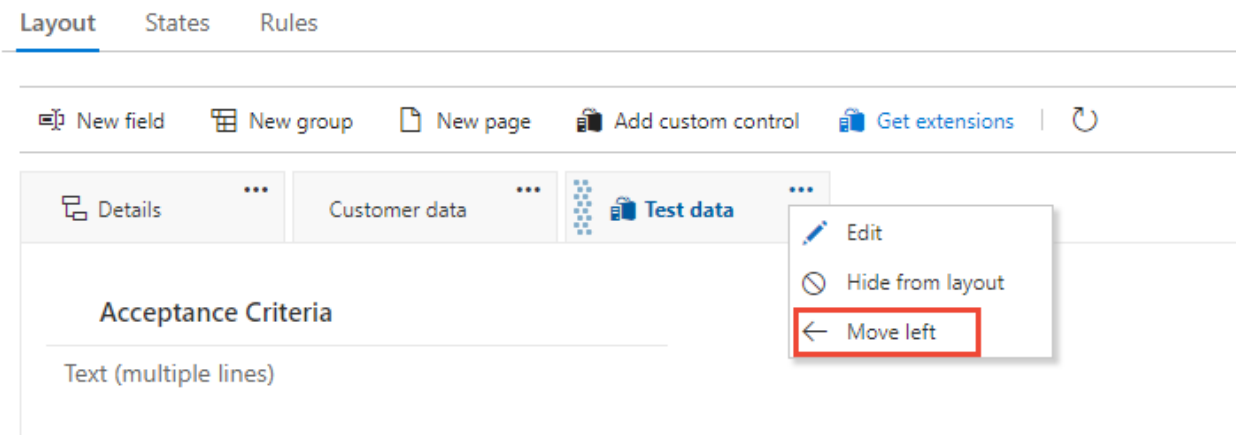
3. Edit the page to add fields or groups.

4. When you've finished adding groups and fields to the page, verify your layout is as expected. Refresh your browser and open a work item of the type you just customized.



Reorder pages

You can reorder custom pages using drag and drop, or choose the move operation from the custom page ... context menu. You can't move the inherited Details page.



Delete a page

Before you delete a page, you'll want to move any fields that your team still uses to another page.

Choose the ✕ Delete option from the custom page ... context menu.

> ⓘ Note

Deleting a page doesn't delete any data. Once data has been defined for a field, it's maintained in the data store and work item history, even if you remove it from the form. You can always choose to re-add the field by choosing **New field** and selecting **Use an existing field**.

Related articles

> **Note**
>
> Review changes made to an inherited process through the audit log. For more information, see Access, export, and filter audit logs.

- Add or modify a custom work item type
- Add and manage fields
- Add extensions and custom controls to a work item type
- Customize a project using an inherited process
- Customize cards on boards

Feedback

Was this page helpful? Yes No

Provide product feedback

Add extensions and custom controls to a work item type (Inheritance process)

Article • 02/24/2023

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

With extensions, you can add rich functionality to your work item forms. An extension comes in four flavors, or contribution types. The following controls appear on the web form layout for all inherited and customizable work item types (WITs):

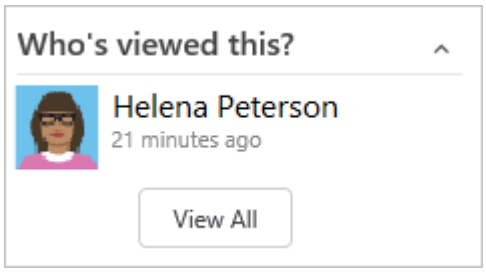
- Field-level contribution (custom controls)
- Group-level contribution
- Page-level contribution

The fourth type, action-level contributions, on the other hand, appear in the context menu of the web form.

> **Important**
>
> This article applies to Azure DevOps Services and Azure DevOps Server 2019 and later versions. To customize any project defined on a collection for TFS 2018 or earlier, see **On-premises XML process model**.

You add all contributions by installing its extension from the Visual Studio Marketplace, Azure DevOps tab. Or, you can create your own custom control.

For example, you can add the **Who recently viewed a work item?** extension. This group-level contribution appears in your work item form as shown.



Prerequisites

Prior to customizing a process, we recommend that you review Configure and customize Azure Boards, which provides guidance on how to customize Azure Boards to meet your

business needs. For a description of the different backlogs and boards, see Tasks supported by Backlogs, Boards, Taskboards, and Plans.

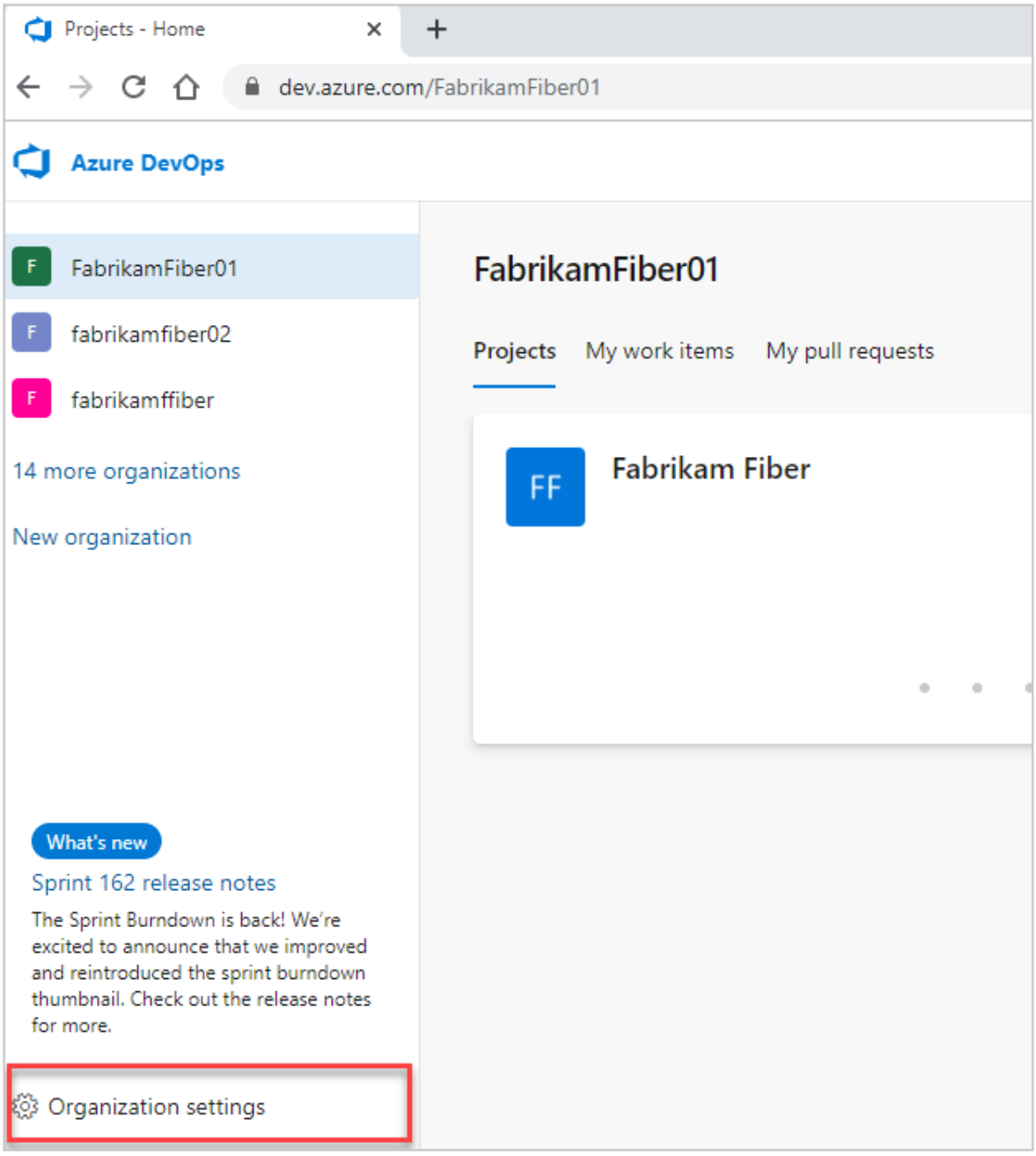
- You must have an organization created in Azure DevOps Services. If you haven't created one yet, do that now.
- To create, edit, and manage processes, you must be a member of the **Project Collection Administrators** group, or have the corresponding collection-level permissions **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**. See Set permissions and access for work tracking, Customize an inherited process.

To add extensions or edit a process, you must be a member of the Project Collection Administrators group or be granted explicit permissions to edit a specific process.

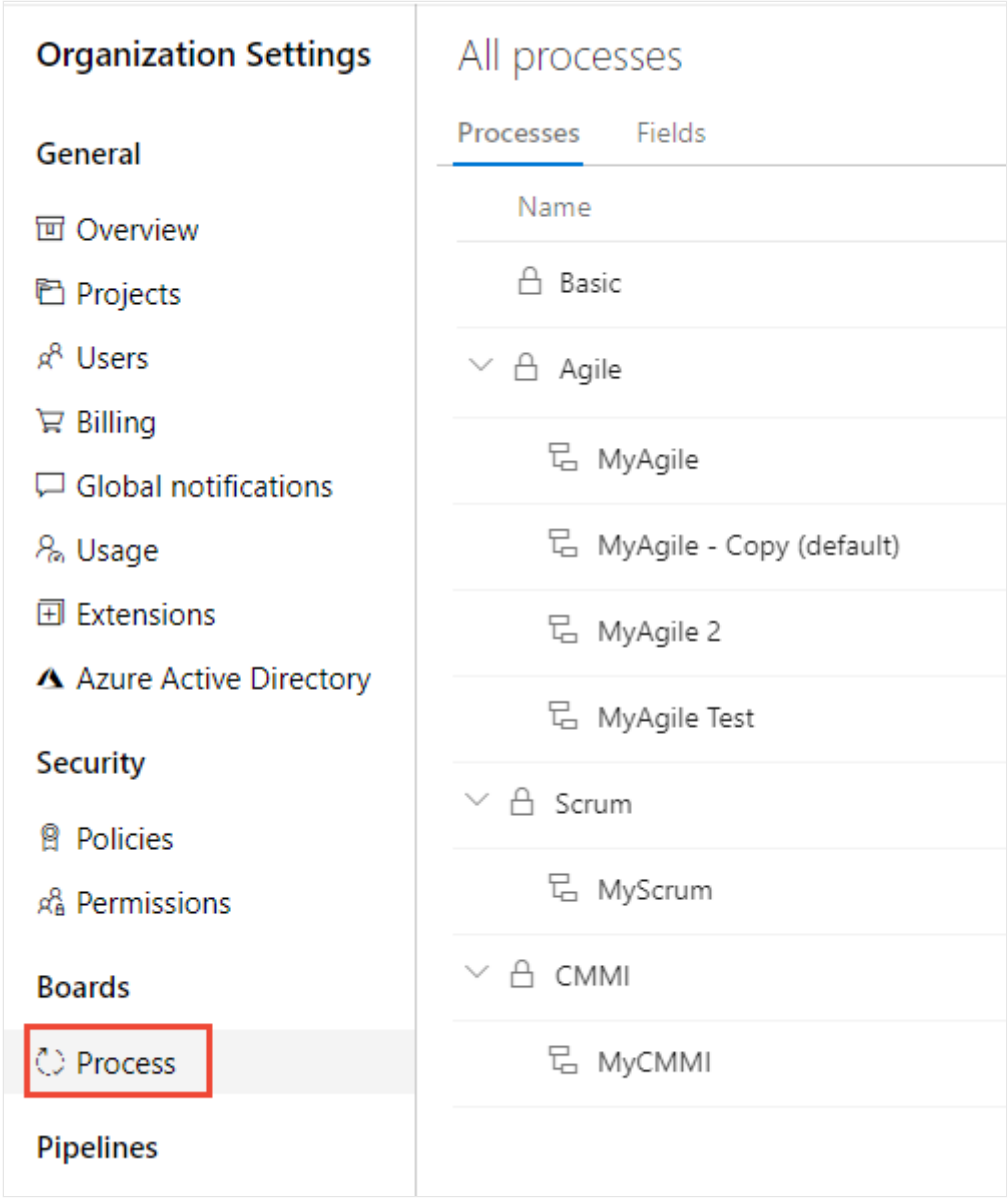
Open Settings>Process

You create, manage, and make customizations to processes from **Organization settings>Process**.

1. Choose the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.



2. Then, choose **Process**.



Important

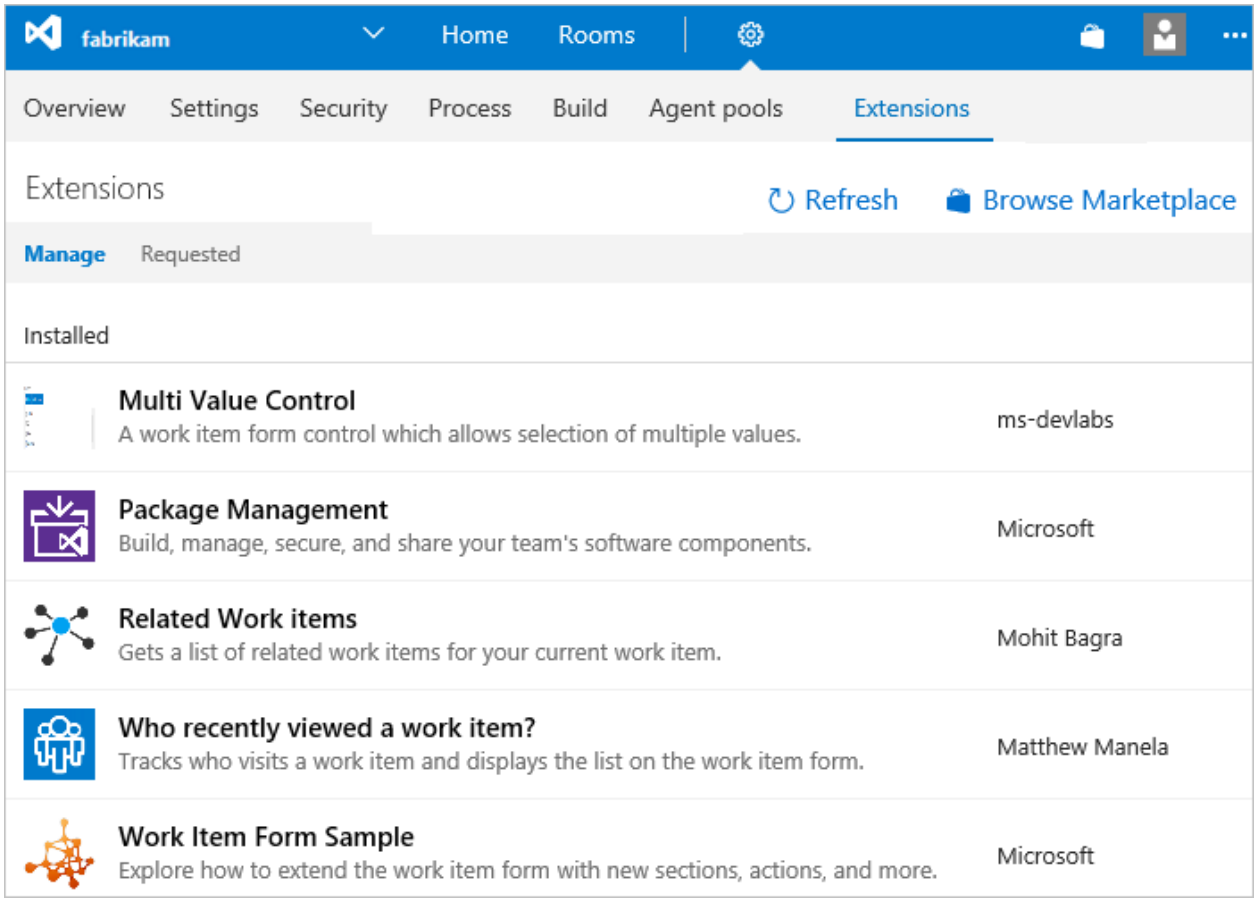
If you don't see **Process**, then you're working from TFS-2018 or earlier version. The **Process** page isn't supported. You must use the features supported for the **On-premises XML process model**.

Note

As you customize an inherited process, all projects that use that process are updated automatically to reflect the customizations. For this reason, we recommend that you create a test process and test project when you have a number of customizations to make in order to test the customizations prior to rolling them out to your organization. To learn more, see **Create and manage inherited processes**.

Review installed Marketplace extensions

From the **Settings** context, **Extensions**, you can view the extensions that are already installed. You may need to choose Refresh to refresh the page to show newly installed extensions.

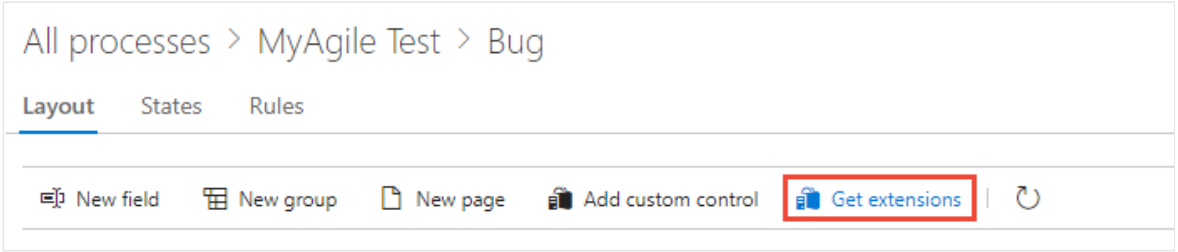


To learn more about extensions, see Install free extensions.

Add a Marketplace extension

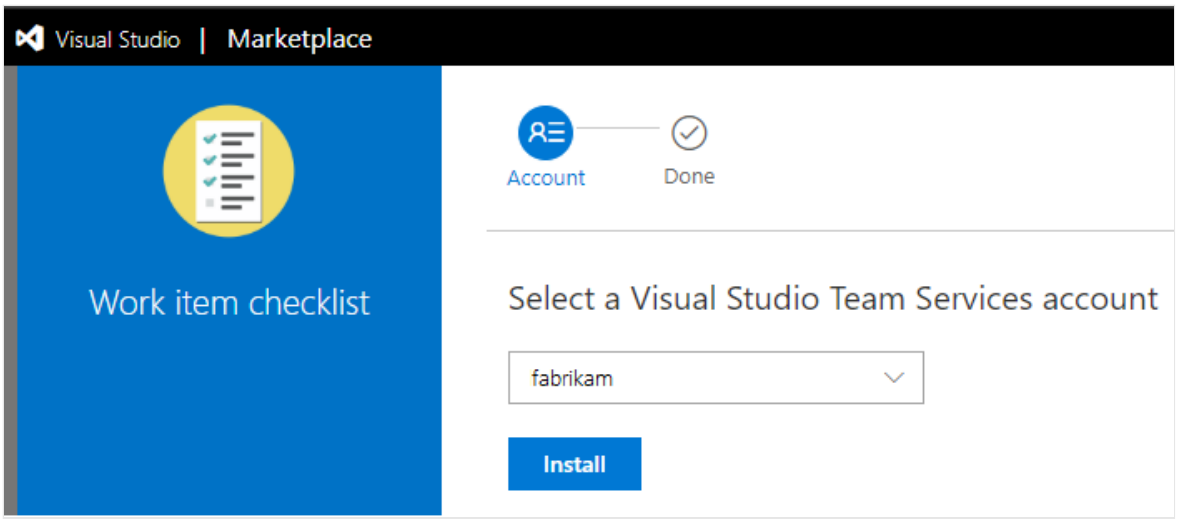
You can browse the Marketplace to determine what extension(s) you want to add.

1. To add an extension to a work item type, choose your inherited process, the WIT, and then choose **Get extensions**.



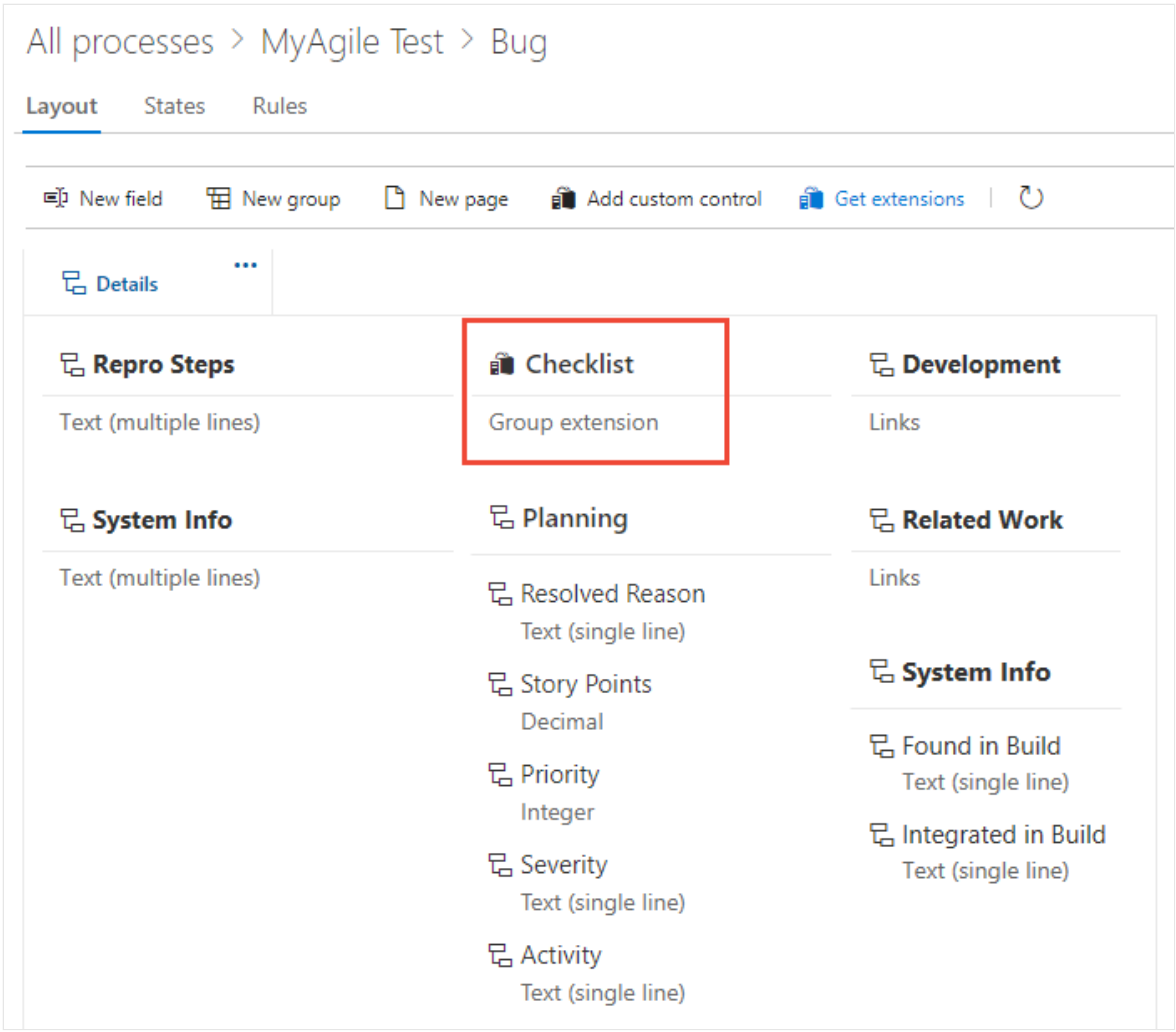
2. Select the extension you want to add. Here we choose the Work item checklist.

3. Select the organization you want to add it to and choose **Install**.



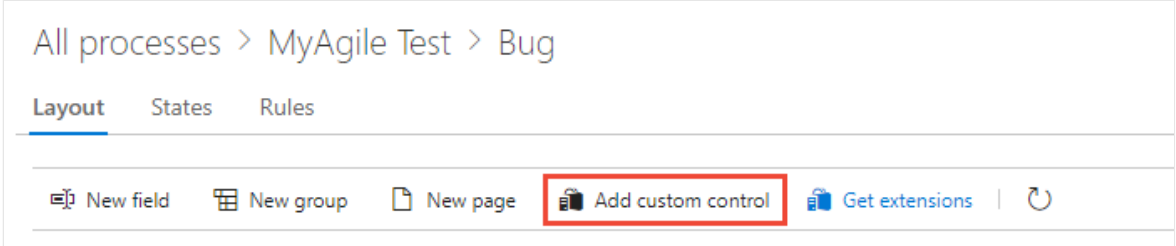
> **Important**
>
> Extensions you install are added to all WITs across all processes.

4. Return to the process and WIT and verify the location of the extension is where you want it. Look for it at the bottom of the middle column and drag it to where you want it on the form.



Add a field-level contribution or custom control

1. Install the custom control as discussed in the previous section.
2. Open **Settings>Work>Process** from a work item form. For details, see Open Settings>Process.
3. (Optional) Add a field to associate with the custom control. Alternatively, you can specify an existing field, inherited or custom.
4. With the WIT selected, choose **Add custom control**.

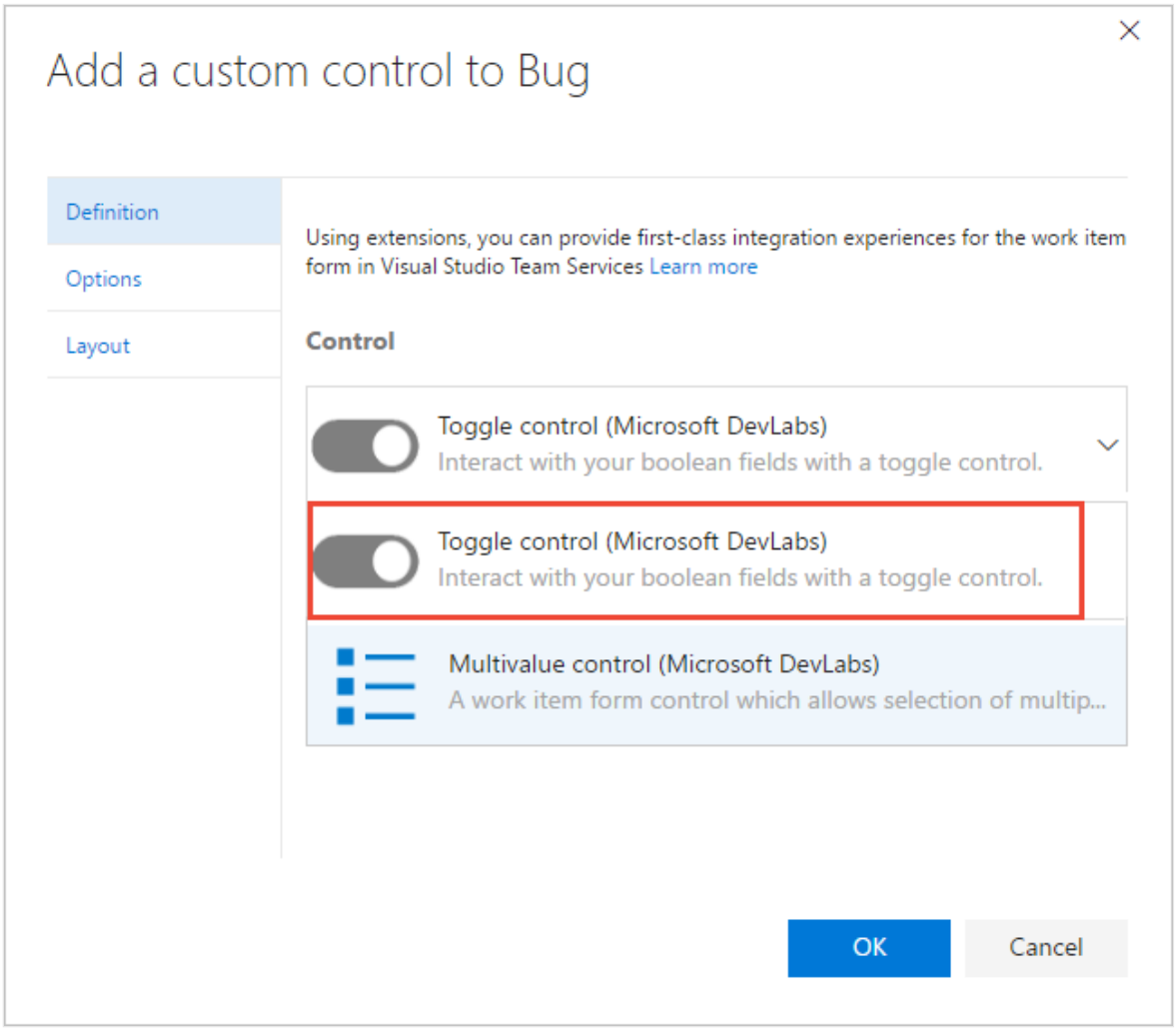


Note

If you don't see the **Add custom control** option, then you haven't installed a field-level extension. Or, you may need to refresh your browser to cause your browser to register any new extensions that have been recently installed.

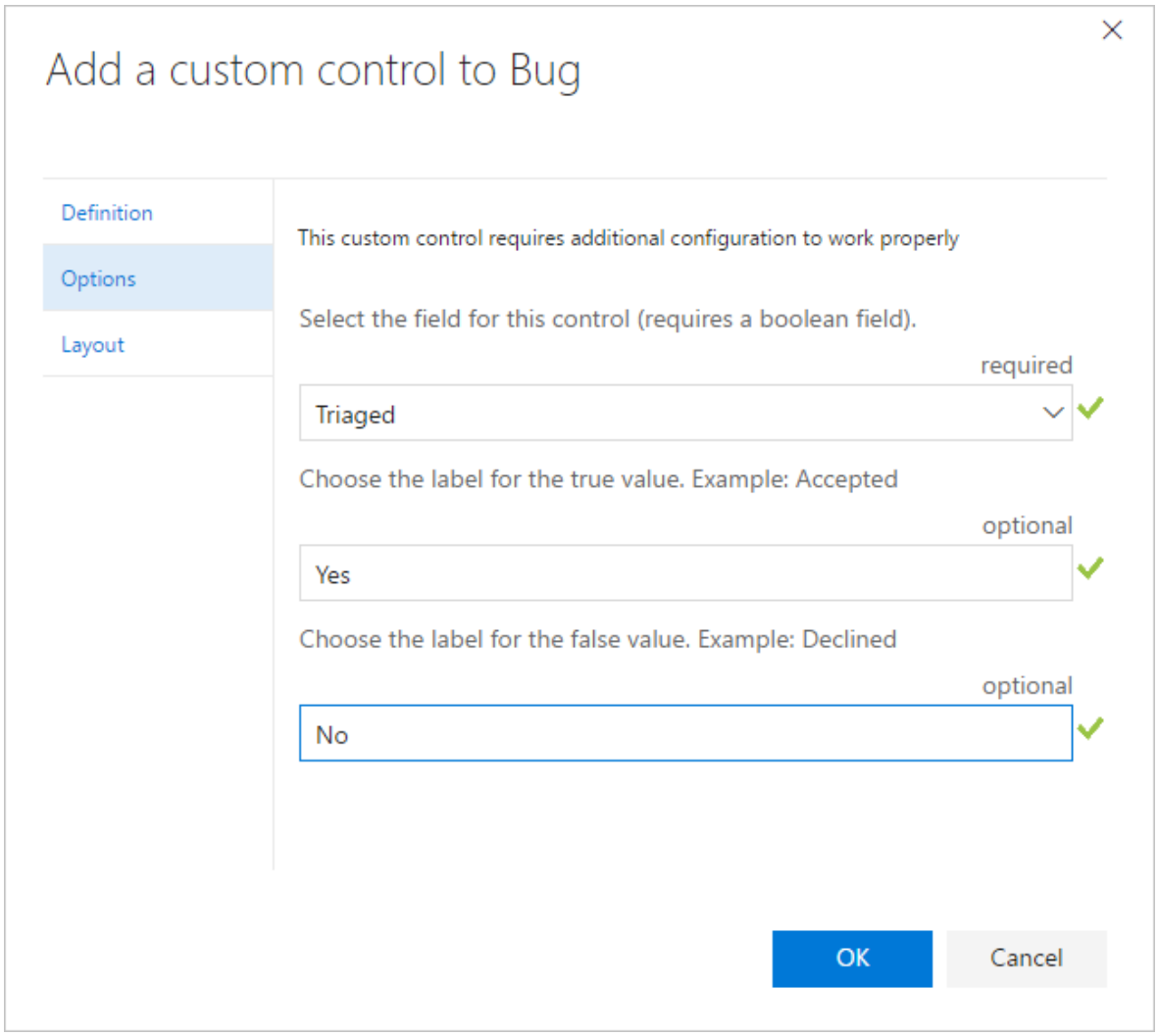
5. Choose the custom control you want from the menu of options.

   For example, here we choose the Toggle control that you can associate with a Boolean field.



6. Choose the **Options** tab and fill out the dialog box. The options you need to specify depend on the custom control you've selected.

   For example, here we specify the custom field, Triaged, and indicate the toggle labels to appear on the form.



7. (Optional) Open the **Layout** tab and specify the label for the field.

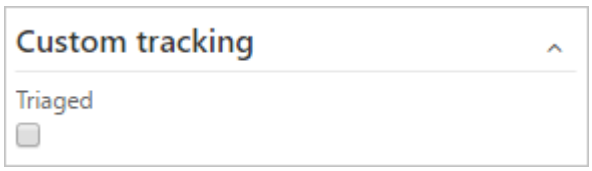
   Fill in any additional required fields. Review the extension description for guidance.

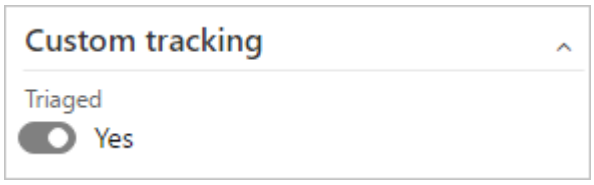
8. Verify the working of the custom control by opening a work item of the type you modified. You may need to refresh your browser to see your changes.

   Here, we show how the control appears by default with the checkbox and then with the toggle control.

Boolean field with checkbox

Boolean field with toggle control





Group-level and page-level contributions

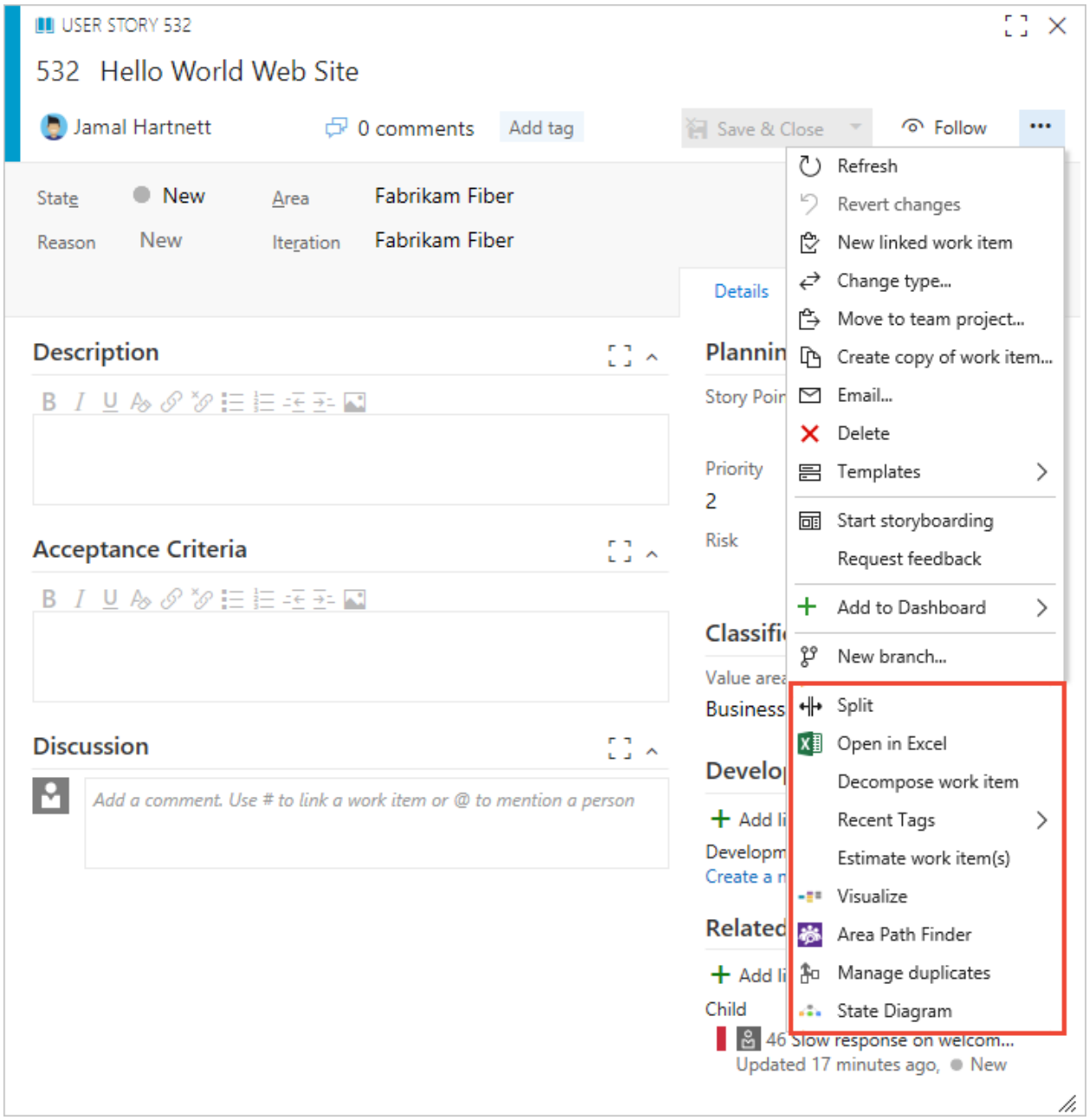
When you add group-level and page-level contributions, they are automatically added to all WITs defined for all processes, both inherited and custom. You can choose to hide these contributions from appearing on the form for a specific WIT, or move it within the form from the default location.

If you've installed a group or page contribution, refresh your browser, and then open a work item to view the placement of the contribution on the form. To move or hide the contribution, follow the guidance provided for the Layout tab.

Action-level contributions

Action-level contributions are added to the context menu of inherited and customizable WITs for both system and inherited processes. All action-level contributions are added to all work item types and cannot be hidden or removed without uninstalling or disabling the extension.

For example, the following image shows nine action-level contributions have been added and appear in the context menu for the user story.



Programmatically add custom controls

You can use these REST APIs to work with extensions:

- Extend the work item form
- Add a custom control to the work item form

Related articles

> **Note**
>
> You can review changes made to an inherited process through the audit log. To learn more, see **Access, export, and filter audit logs**.

- Add a custom field to a work item type
- Customize a work item type
- Customize the web layout
- Customize a project using an inherited process

Rules and rule evaluation

Article • 03/25/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Rules are used to set or restrict value assignments to a work item field. There are two main types of rules, auto-generated rules and custom rules defined for a process or project. Auto-generated rules minimize the need to add custom rules for areas that should work in a standard way.

You define custom rules in order to support your business use cases. Depending on a field's data type, you can set various restrictions on what data can be entered into that field. You can specify values for a pick list (drop-down menu), set default values, clear entries, or restrict changes. With conditional rules, you can apply rules to a field based on dependencies between different fields' values. You can also restrict who can modify a field or scope a rule to only apply to a group.

Read this article to understand the following:

- How the system applies auto-generated rules
- Restrictions placed on definition of custom rules on system fields
- The different types of custom rules you can apply
- How rules are evaluated
- Difference between rules defined for an Inheritance process versus an On-premises XML process
- Why you should minimize the number of custom rules you define

Prior to defining custom rules, read Configure and customize Azure Boards to gain a broad understanding of how to customize Azure Boards to meet your business needs.

> **Tip**
>
> Minimize the number of rules you define for a WIT. While you can create multiple rules for a WIT, addition rules can negatively impact performance when a user adds and modifies work items. When users save work items, the system validates all rules associated with the fields for its work item type. Under certain conditions, the rule validation expression is too complex for SQL to evaluate.

Auto-generated rules

Auto-generated rules minimize the need to add custom rules for areas that should work in a standard way.

State transition rules

Inherited processes generate the entire set of any-to-any state transition rules dynamically for each custom work item type and custom state added to a workflow. A transition from any state to any state is valid.

For On-premises XML processes, you must specify the valid transitions within the `WORKFLOW` section of the work item type definition.

State transitions and By/Date field rules

By/Date fields correspond to **Created By/Date**, **Activated By/Date**, **Resolved By/Date**, and **Closed By/Date**.

For inherited processes, these fields are automatically set or cleared when you transition a work item from one state to another. The Changed By/Date fields aren't included as they are updated with each work item save and are unrelated to state transitions.

Default rules and behaviors that govern these fields include:

1. The *Closed* state is always contained in the *Completed* state category.
2. The *Completed* state category is not configurable, and is associated with one and only one State.
3. This *Closed* state is always *Closed* for Agile and CMMI processes, and always *Done* for Scrum and Basic processes.
4. Auto-generation of these rules are affected by locale as the rule condition contains the State name, which is localized. The system generates different rules for different locales.
5. Auto-generated rules for these fields are only specified for work item types that include these fields. It is possible for a work item type to not include one or more of these fields.
6. These rules are needed when a work item type has custom states, or the work item type is a custom work item type.
7. These rules only apply to inherited processes; they are never generated for the Hosted XML or On-premises XML processes.

Workflow states are associated with state categories to support the workflow on Kanban boards. To learn more, see How workflow states and state categories are used in Backlogs and Boards.

State Change Date field rules

These rules are technically a lot simpler than Closed By/Closed Date rules because they are not dependent on any particular state. For any work item type, the same rules will always work. They need to be auto-generated because some OOB work item types do not contain the State Change Date field, so when the user adds this field to a custom work item type, these rules need to be auto-generated as well. The same principles for Closed By/Closed Date rules apply here as well.

Custom rules

All custom rules are optional. For an inherited process, you specify a rule which consists of a condition plus action. For an On-premises XML process, you specify rules for a field or within the workflow.

There isn't a one-to-one mapping between the two processes. In some cases, the XML element rule is defined within the **Edit field** dialog for the inherited process and not as a rule. Other XML elements, such as `FROZEN`, `MATCH`, `NOTSAMEAS`, aren't supported in the inherited process.

Note the following:

- Rules are always enforced, not only when you are interacting with the form but also when interfacing through other tools. For example, setting a field as read-only not only applies the rule on the work item form, but also through the API and Excel Azure DevOps Server Add-in.
- Inherited process entries specify conditions and actions to make a complete rule. XML elements don't make those distinctions.
- Field rules don't support assigning values that are the sum of two other fields or performing other- mathematical calculations. However, you may find a solution that fits your needs via the TFS Aggregator (Web Service) Marketplace extension. See also Rollup of work and other fields.
- You may find additional solutions to applying custom rules to fields using a Marketplace extensions, such as the Work item form control library extension.

Rule composition

Inheritance process

For an inherited process, each rule consists of two parts: Conditions and Actions. Conditions define the circumstances which must be met in order for the rule to be

applied. Actions define the operations to perform. For most rules, you can specify a maximum of two conditions and 10 actions per rule. All custom rules require all conditions to be met in order to be run.

As an example, you can make a field required based on the value assigned to the state and another field. For example:

`(Condition) When a work item State is` *Active*
`(Condition) And when the value of` *Value Area = Business*
`(Action) Then make required` *Story Points*

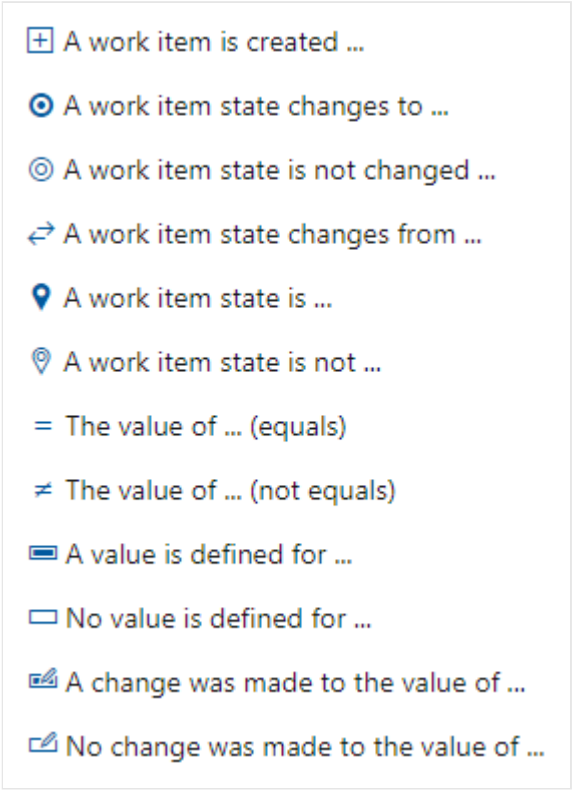
> **Note**
>
> Currently, only one condition is supported for state-transition rules. If you're applying rules based on State, see Apply rules to workflow states.

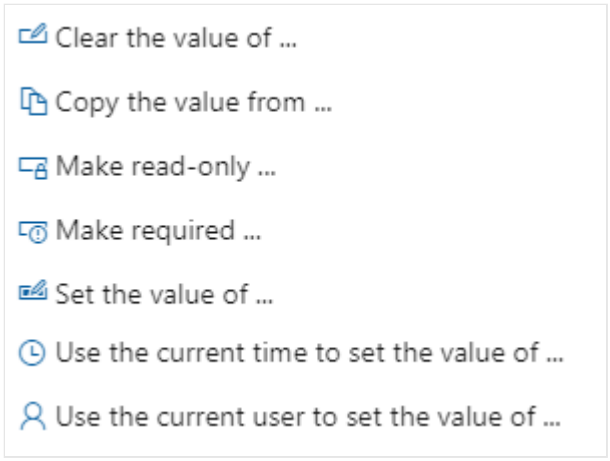
The following table summaries the Actions that are available with the selected Conditions.

Condition

Supported Actions

Set field value or make required or read-only

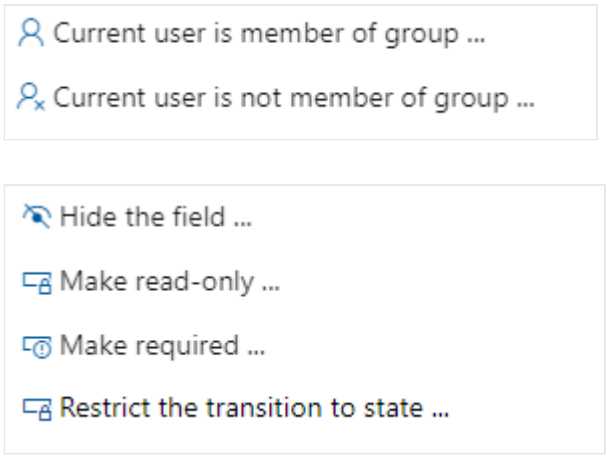




Restrict a transition based on State



Based on and user or group membership, set field attribute or restrict a State transition



What happens if too many rules are defined

A single SQL expression is defined per project to validate work items whenever they are created or updated. This expression grows with the number of rules you specify for all work item types defined for the project. Each behavioral qualifier specified for a field results in an increase in the number of sub-expressions. Nested rules, rules that apply only on a transition or conditioned on the value of some other field, cause more conditions to be added to an `IF` statement. Once the expression reaches a certain size

or complexity, SQL can't evaluate it any more and generates an error. Removing some WITs or eliminating some rules, can resolve the error.

You can specify values for a pick list (drop-down menu), set default values, clear entries, or restrict changes. With conditional rules, you can apply rules to a field based on dependencies between different fields' values. You can also restrict who can modify a field or scope a rule to only apply to a group.

Work item rules do not exist as a single collection. The rules are actually dynamically generated and merged from different data sources. The merge logic is a simple one, consolidate identical rules, but don't trim conflicting rules.

Bypass rules

In general, all work items are validated by the rule engine when users modify the work item. However, to support certain scenarios, users assigned the **Bypass rules on work item updates** project-level permission can save work items without rules being evaluated.

Rules can be bypassed in one of two ways. The first is through the Work Items - update REST API and setting the `bypassRules` parameter to `true`. The second is through the client object model, by initializing in bypassrules mode (initialize `WorkItemStore` with `WorkItemStoreFlags.BypassRules`).

System fields and custom rules

System fields have System.*Name* reference names, for example **System.Title** and **System.State**.

The following system fields are required to have a value: **Area ID**, **Changed Date**, **Created Date**, **Created By**, **State**, and **Reason**.

The rule engine restricts setting conditions or actions to system fields except as follows:

- You can make **State** and **Reason** fields read-only.
- You can apply most rules to the **Title**, **Assigned To**, **Description**, and **Changed By** fields.

If you don't see a field listed in the drop-down menu of the rule user interface for the Inheritance process, this is why. For example, if you try to make **Area Path** (System.AreaPath) read-only based on a condition, the Area Path field isn't available for

selection. Even if you're able to specify a system field, the rule engine may restrict you from saving the rule.

Default and copy rules

Default and copy rules modify the values of work item fields. They define run-time behavior and constraints, such as specifying default values, clearing fields, requiring fields to be defined, and more.

You can restrict application of these rules based on the current user's group membership as described in User or group membership rule restrictions.

Inheritance process

Most of these rule actions can be applied with the selection of any condition.

Inherited process action	Description
`Copy the value from...`	Specifies another field that contains a value to be copied into the current field.
`Clear the value of...`	Clears the field of any value that it contains.
`Use the current time to set the value of ...`	Sets the time for a field based on the current user's time setting.

Constraint rules

Constraint rules restrict changing the value of a field. They define the valid states for a work item. Each constraint operates on a single field. Constraints are evaluated on the server on work item save, and if any constraint is violated the save operation is rejected.

You can restrict application of these rules based on the current user's group membership as described in User or group membership rule restrictions.

Inheritance process

Most of these rule actions can be applied with the selection of any condition.

Inherited process action

Description

`Hide the field...`
Only available when a group membership condition is selected.

Specifies to not show the field on the work item form, essentially removing the ability for the current user to change the field's value.

`Make read-only`

Prevents a field from being modified at all. You might want to apply this rule under certain conditions. For example, after a work item is closed, you want to make a field read-only to preserve the data for reporting purposes.
To specify the field default is read-only, specify in Edit field dialog, **Options** tab.

`Make required`

Requires a user to specify a value for the field. Users cannot save a work item until they have assigned values to all required fields.
To specify the field default is required, specify in Edit field dialog, **Options** tab.

Pick lists

Pick lists define the values that a user can or can't choose for a String or Integer field. Values defined in a pick list appear on a work item form and the query editor.

Inheritance process

For an Inherited process, pick lists are defined through the Edit field dialog.

Edit field dialog

Description

Definition tab for a picklist field

Defines a list of allowed values for the field. Allowed values are values that are available for selection in a field list on work item forms and in the query builder. You must select from one of these values.

Check the **Allow users to enter their own values** checkbox within the **Options** tab to allow users to specify their own entries

Defines a list of suggested values for the field. Suggested values are values that are available for selection in a field list on work item forms and in the query builder. You can enter other values additionally to the ones in the list.

Conditional field values or changes

Conditional rules specify an action based on the value of a field equaling or not equaling a specific value, or if a change was or wasn't made to the value of a specific field. In general, conditional rules are applied first over unconditional rules. When multiple conditional rules evaluate to true, the order of execution is: When, WhenNot, WhenChanged, WhenNotChanged.

Inheritance process

You can specify multiple conditional rules per field. However, you can only specify a single driving field per conditional rule.

Inherited condition

Description

`The value of ... (equals)` [When]

Specifies one or more rules to apply to the current field when another field has a specific value.

`A change was made to the value of ...` [WhenChanged]

Applies one or more rules to the current field when a specific field's value is changed.

`The value of ... (not equals)` [WhenNot]

Applies one or more rules to the current field when another field does not have a specific value.

`No change was made to the value of ...` [WhenNotChanged]

Applies one or more rules to the current field when a specific field's value is not changed.

Inherited action

Description

`Clear the value of ...`
`Copy the value from ...`
`Make read-only ...`
`Make required ...`
`Set the value of ...`
`Use the current time to set the value of ...`
`Use the current user to set the value of ...`

Specifies the action to take on a specific field.

User or group membership rule restrictions

You can restrict application of a rule based on the current user's membership. We recommend you scope the rule to an Azure DevOps security group, and not a single user, although you can specify the latter. To have the rule scoped to multiple groups, you must create a parent Azure DevOps group that includes the set of groups that you want to use.

Process implementation

Tip

To avoid rule evaluation issues that may arise, specify Azure DevOps security groups and not Microsoft Entra ID or Active Directory security groups. To learn

more, see **Default rules and the rule engine**.

Inheritance process

As indicated in the following table, to restrict a rule based on the current user's membership, you specify one of two conditions for an Inherited process. These rules are active for Azure DevOps 2020 and later versions.

Applies to

Rule

Condition

`Current user is a member of group ...`

`Current user is not member of group ...`

Action

`Hide the field ...`

`Make read-only ...`

`Make required ...`

`Restrict the transition to state ...`

Use tokens to reference users or groups

Identity or people-picker fields can accept values that reference both users and groups. When you restrict a rule to a group, you indicate the domain or scope of the group. For some values, you can use tokens.

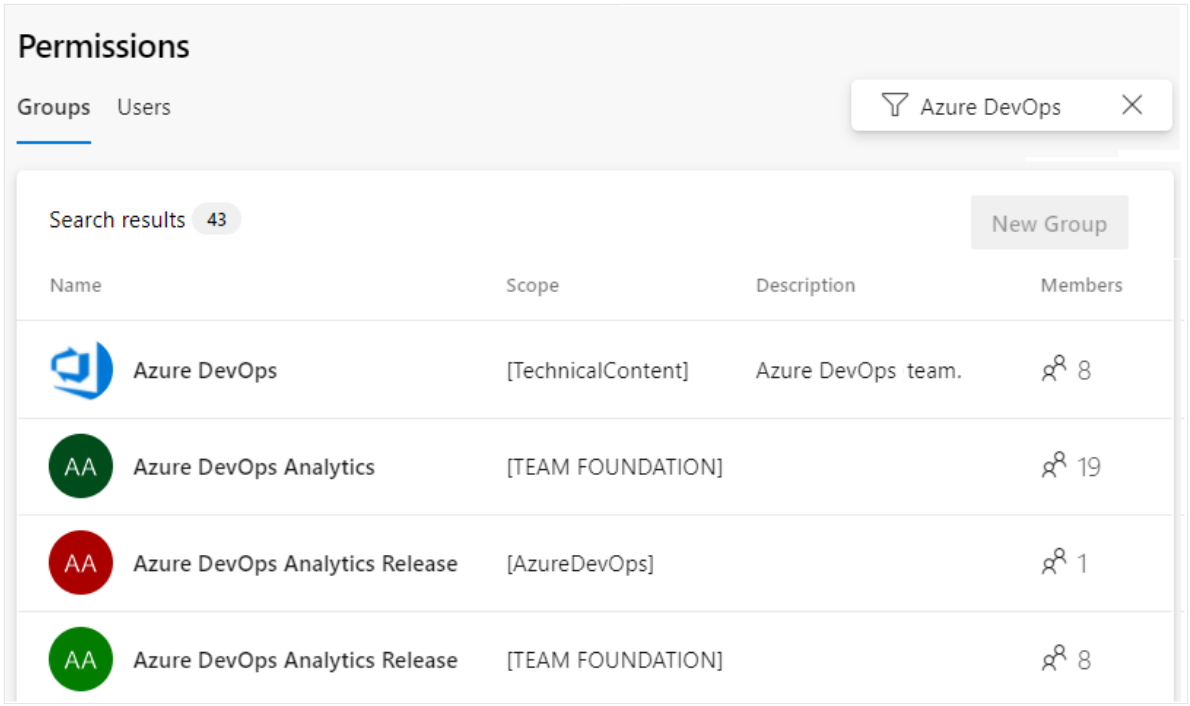
Inheritance process

Examples of tokens include the following:

- [*ProjectName*], such as [Fabrikam], [FabrikamFiber], [MyProject]
- [*OrganizationName*], such as [fabrikam], [myorganization]
- [*CollectionName*], such as [fabrikam], [myorganization]

To learn about the scopes available for your project or organization, go to the **Project Settings**>**Permissions**>**Groups** or **Organization**

Settings>Permissions>Groups page, you can filter the list as needed. For example, the following image shows the first four entries to a filtered list based on *Azure DevOps*. To learn more, see Change project-level permissions or Change project collection-level permissions.



To learn more about default security groups, see Permissions and groups

Rule evaluation

Rules that specify a condition based on the user or group membership of the user modifying a work item are evaluated in one of two ways. When the rule is evaluated, the application needs to determine whether the rule applies to the current user by checking if that user is or isn't a member of the specified group.

- When modifying the work item from the web portal, REST API, or **azure boards** command, a request to the Microsoft Entra ID or Active Directory is made. No problems occur for this operation.
- When modifying the work item from Visual Studio, Excel or other custom tool using the WIT Client Object Model, the request to evaluate membership is based on a client cache. The client cache is not aware of Active Directory groups.

> **Note**
>
> Visual Studio 2019 Team Explorer for projects using GIT has been re-written to use REST APIs.

To avoid problems with users updating work items from various clients, specify Azure DevOps security groups instead of Active Directory groups. You can easily create an Azure DevOps security group to correspond to an Active Directory group. To learn how, see Add or remove users or groups, manage security groups.

> **Note**
>
> The WIT Client OM is deprecated. As of January 1, 2020, it no longer is supported when working against Azure DevOps Services and Azure DevOps Server 2020.

Order in which rules are evaluated

Rules are typically processed in the sequence in which they are listed. However, the complete sequence for evaluation of all rules isn't fully deterministic.

This section describes the expected behavior and interactions when you apply conditional, copy, and default rules.

Inheritance process

The following steps show, in the correct sequence, the interactions that Azure DevOps performs and by the user of a work-item form. Only steps 1, 8, and 13 are performed by the user.

1. From an Azure DevOps client--such as the web portal or Visual Studio Team Explorer--a user creates a new work item or edits an existing work item.
2. Fill in field defaults. For all fields, apply any defaults assigned to the field that aren't part of a conditional clause.
3. Copy or set field values. For all fields, apply any rules to copy a value or set the value of a field that aren't part of a conditional clause.
4. For all fields with a When conditional rule that matches, apply rules to set or copy a field value.
5. For all fields with a When Not conditional rule that matches, apply rules to set or copy a field value.

   The system always processes **When** rules before **When Not** rules.
6. For all fields that have had their values changed since step 1 and that contain **When Changed** rules, apply rules to set or copy a field value.

7. Allow the user to start editing.
8. The user changes a field value and then moves focus from the field.
9. Process any **When** rules for that field that match the new value.
10. Process any **When Not** rules for that field that match the new value.
11. Process any **When Changed** rules for that field that match the new value.
12. Return editing ability to the user.
13. The user saves the changes to the data store.
14. For all fields, apply any `Use the current time to set the value of ...` actions that are defined for the field either directly or indirectly under a conditional rule.

Related articles

- Sample rule scenarios
- Work item field index
- Work item fields and attributes
- Change project-level permissions
- Change project collection-level permissions
- Permissions and groups
- **Inherited process**
  - Add a custom rule to a work item type
  - Apply rules to workflow states
- **On-premises XML process**
  - Define a default value or copy a value to a field
  - Define pick lists
  - Assign conditional-based values and rules

Feedback

Was this page helpful? Yes No

Provide product feedback

Sample custom rule scenarios

Article • 12/06/2023

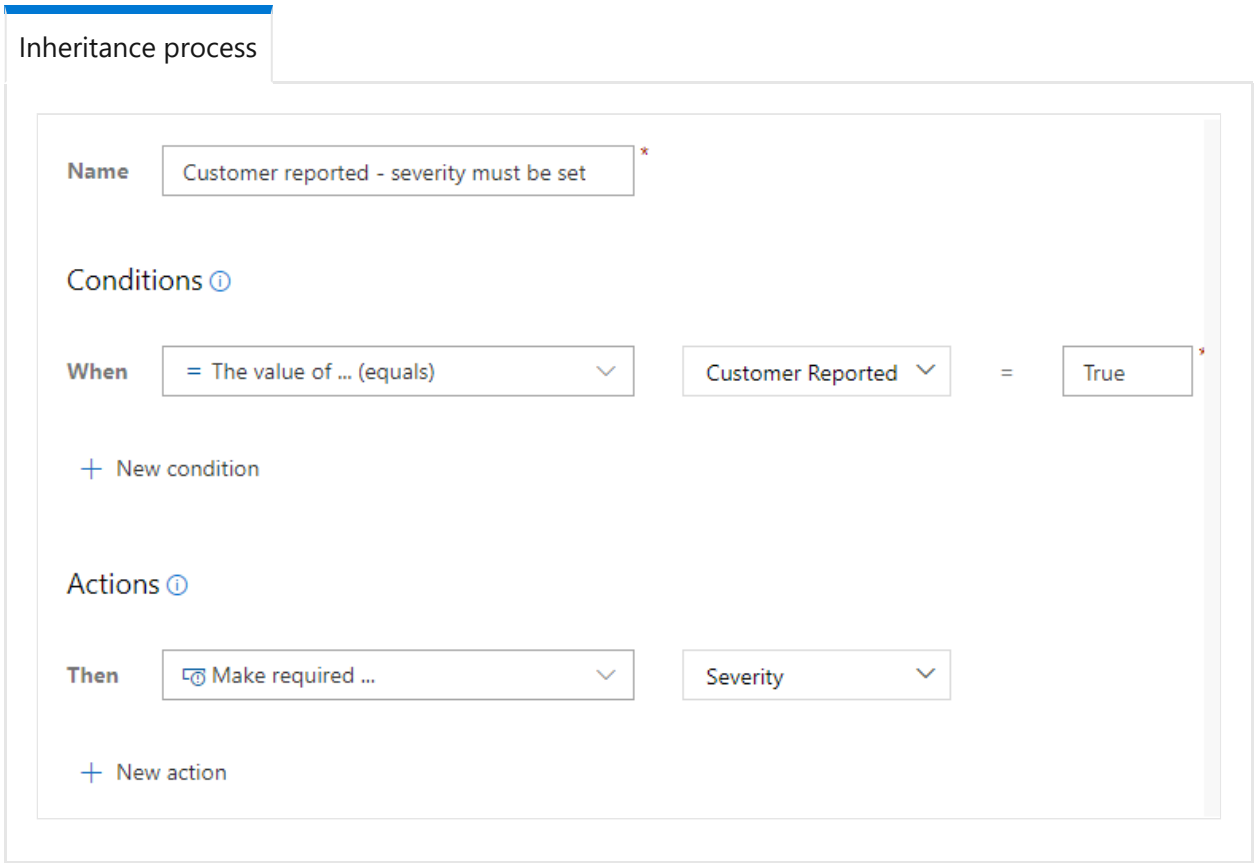
Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

This article provides examples of custom rule definitions. All custom rules are defined for a work item type. Examples are provided for both the Inherited and On-premises XML process models.

Prior to adding custom rules, read Rules and rule evaluation and Add a rule to a work item type (Inheritance process).

Define a dependent required field

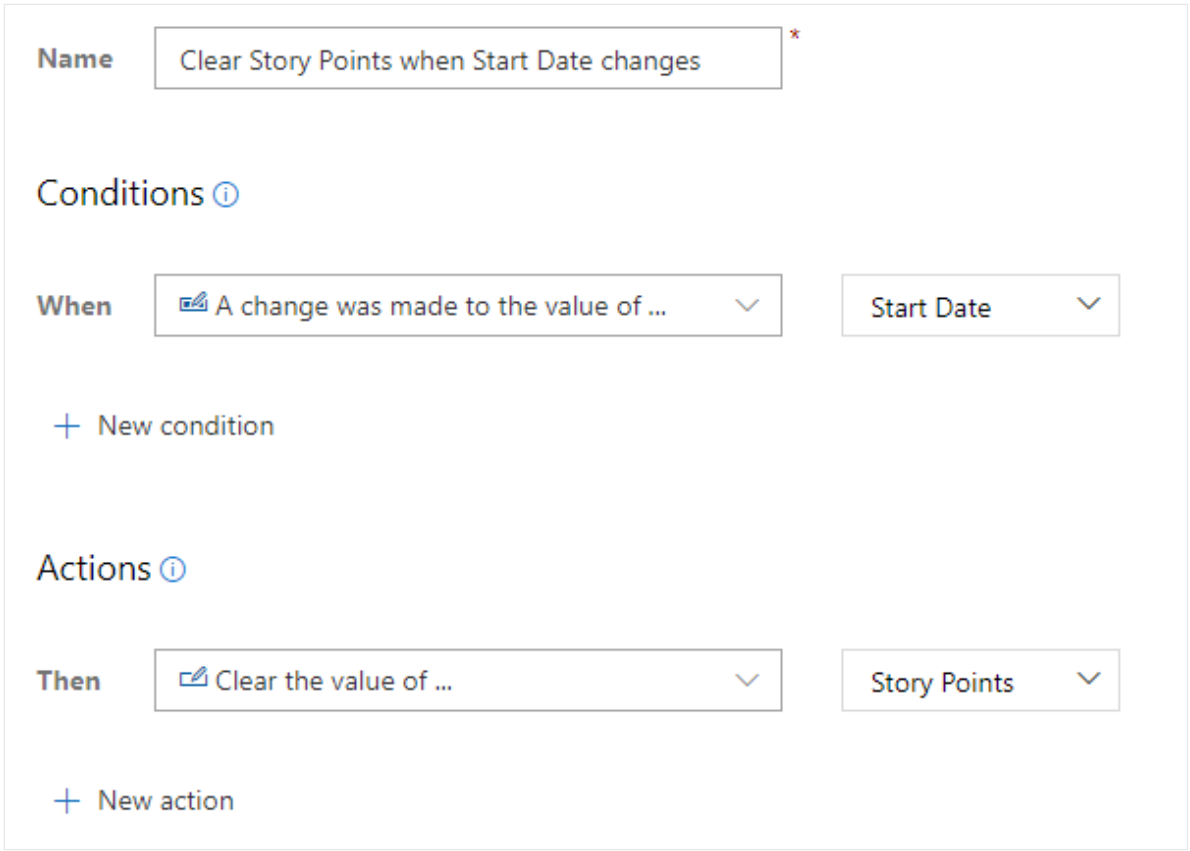
You can specify that a field is required only when another field contains a specific value. In the following example, when a customer reports an issue, the custom **Customer Reported** field is set to *True*, and the **Severity** field becomes required. If the issue isn't reported by a customer, a value for the **Severity** field isn't required.



Clear the value of a dependent field

The following example illustrates defining a custom rule to clear the value for **Story Points** when a change is made to the **Start Date**.

Inheritance process



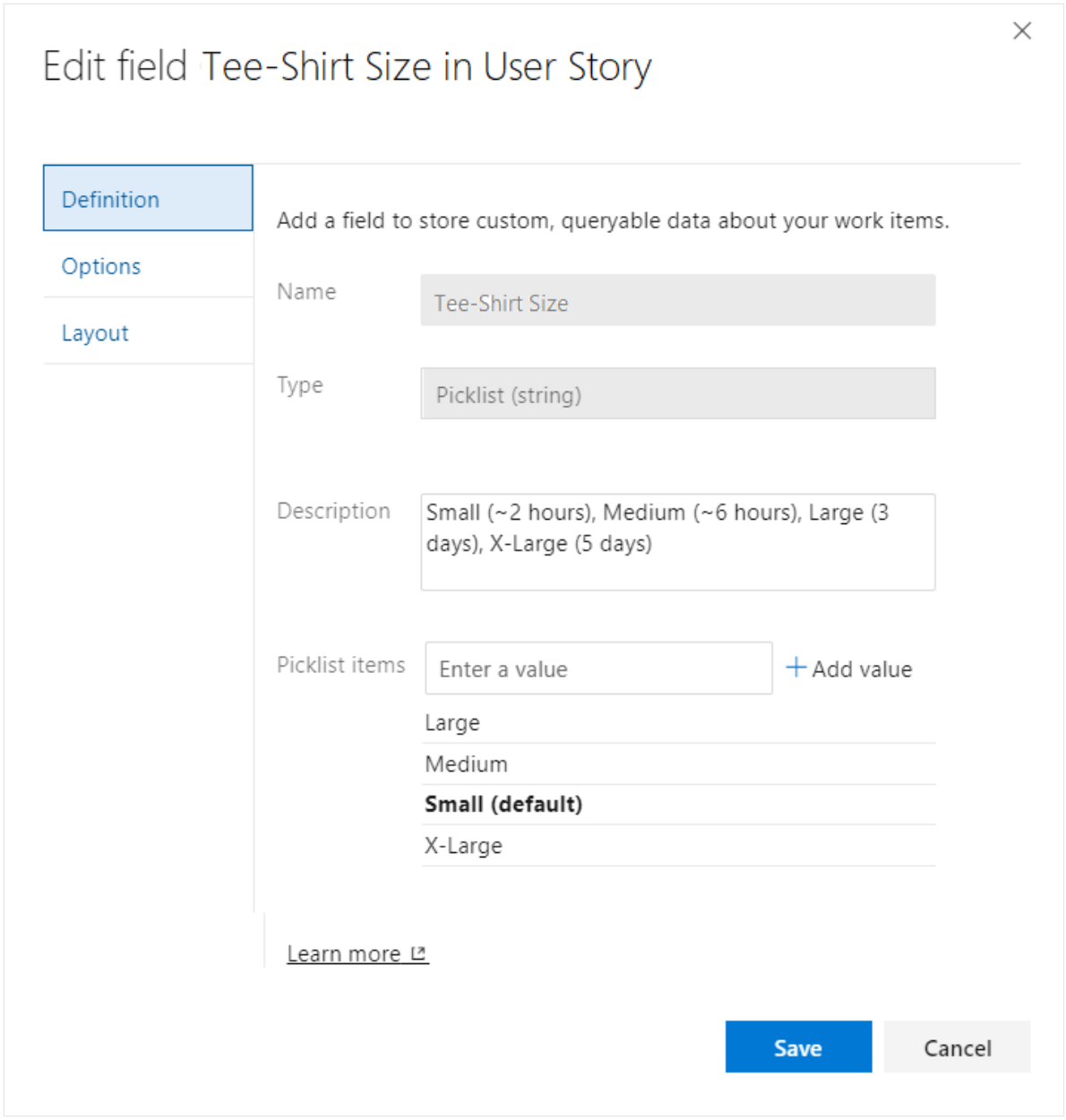
Set a dependent field value

The following examples illustrate how to map the values of the **Size** field depending on the value selected for the custom field, **Tee-Shirt Size** field.

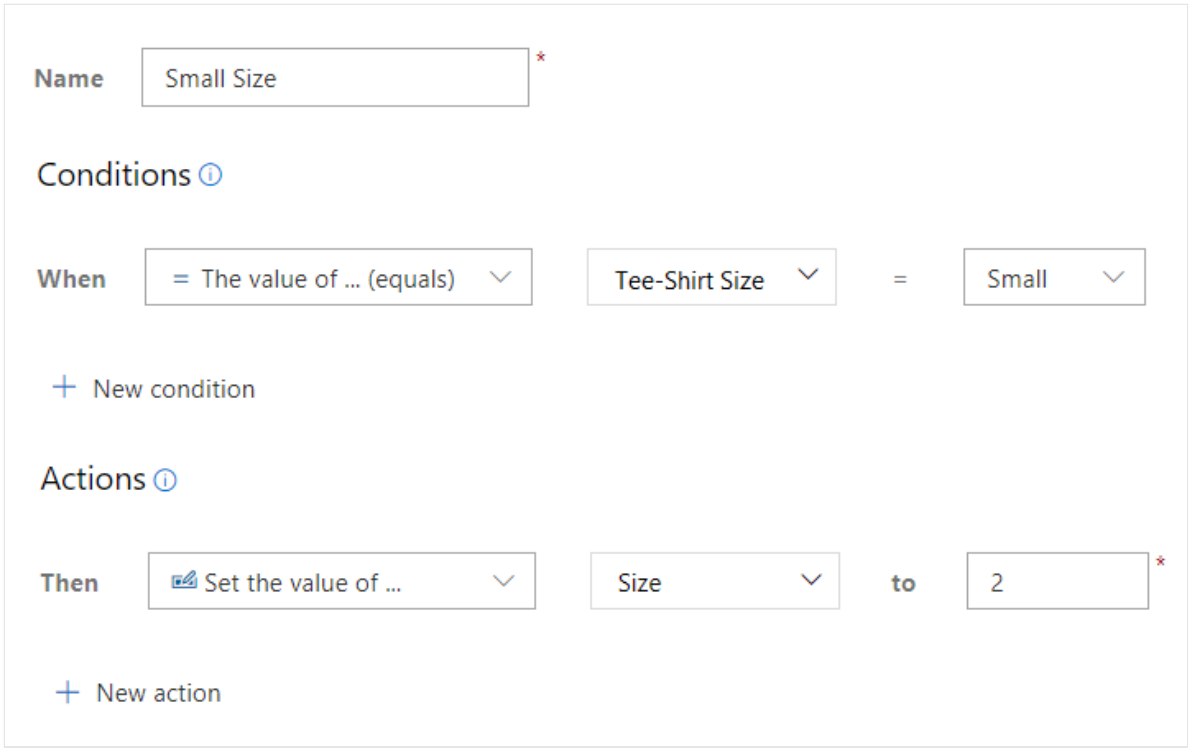
Inheritance process

The **Tee-Shirt Size** pick-list consists of four values *Small*, *Medium*, *Large*, and *X-Large*. Four custom rules are defined to assign the **Size** field when the **Tee-Shirt Size** field is changed to a specific value. To simplify usage, the default value of the **Tee-Shirt Size** is *Small*.

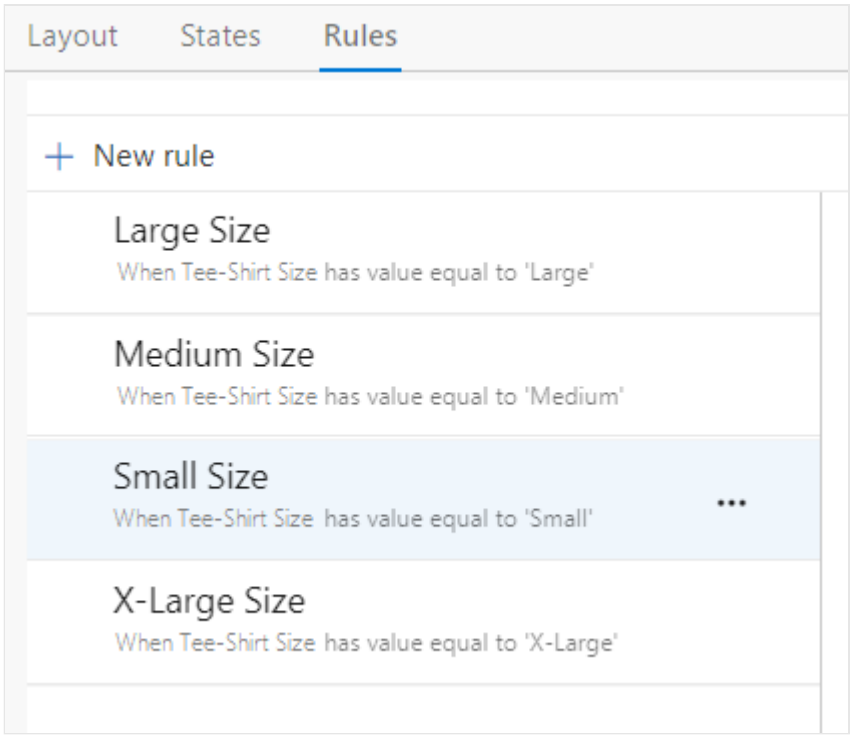
Edit field dialog for Tee-Shirt Size field


Edit field Tee-Shirt Size in User Story
Definition
Options
Layout
Add a field to store custom, queryable data about your work items.
Name
Tee-Shirt Size
Type
Picklist (string)
Description
Small (~2 hours), Medium (~6 hours), Large (3 days), X-Large (5 days)
Picklist items
Enter a value
Add value
Large
Medium
Small (default)
X-Large
Learn more
Save
Cancel

Custom rule



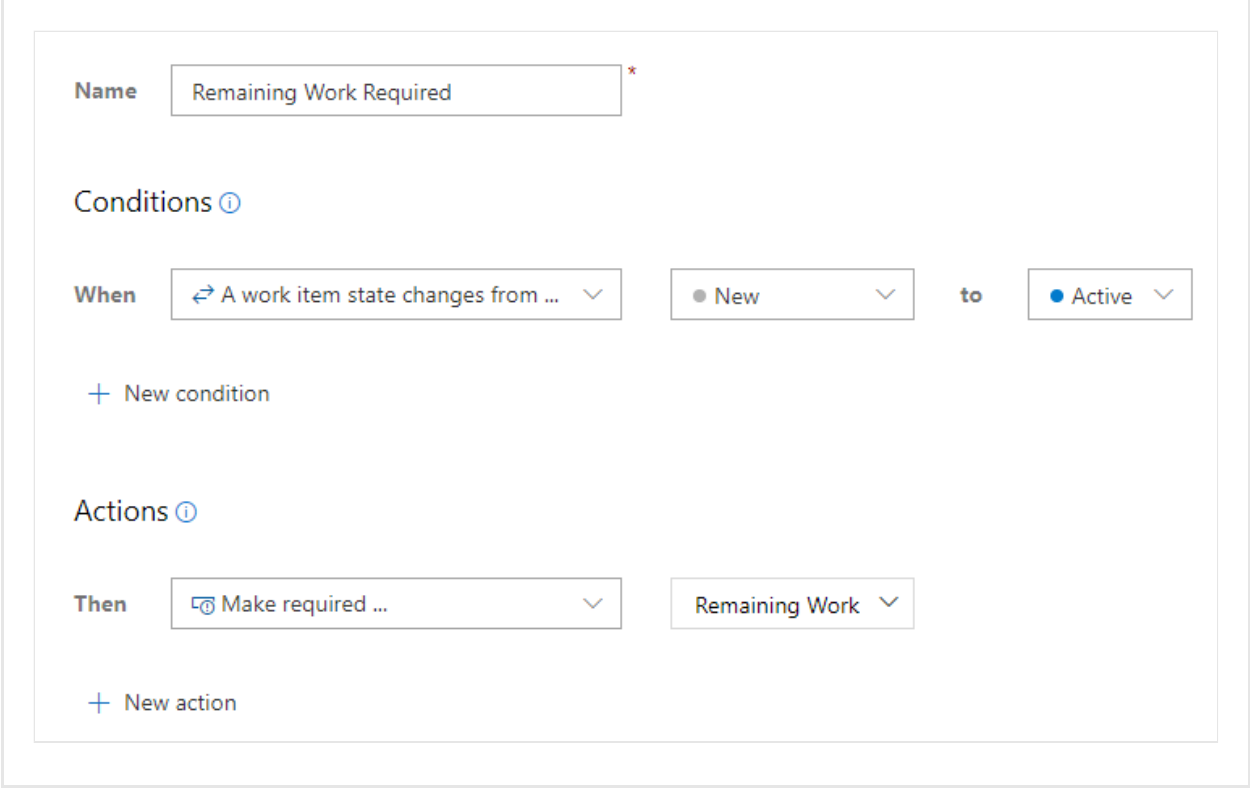
Four custom rules



Require a field value upon State changes

The following example shows how you can require specification of the **Remaining Work** field when the task workflow **State** changes to *Active*.

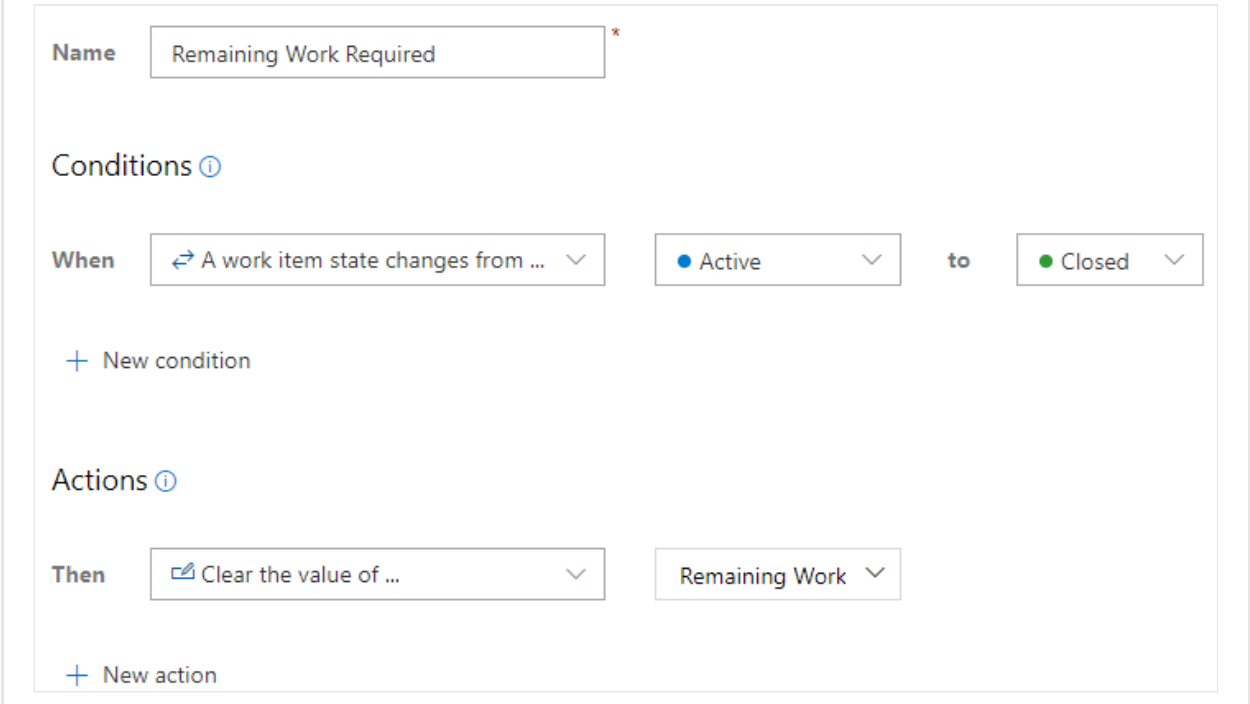
Inheritance process



Clear the value of a field upon close State

To automate clearing the **Remaining Work** field upon closing a task, define a custom rule as indicated.

Inheritance process

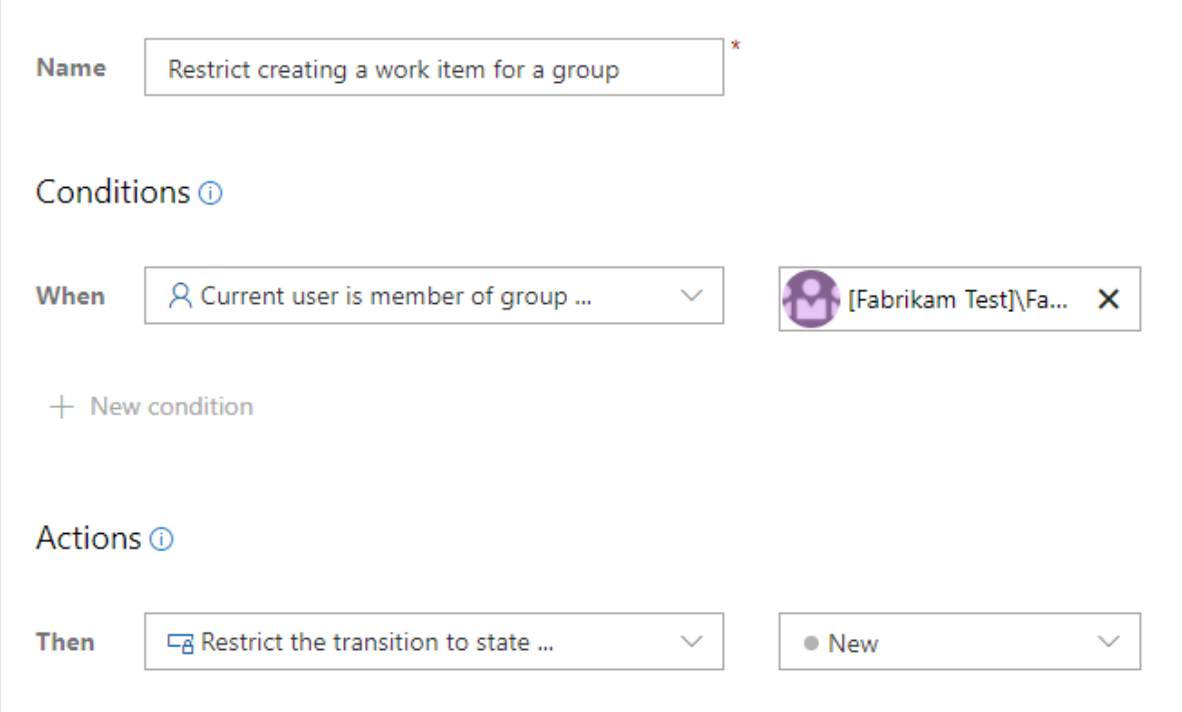


Restrict creation of work items by a group

A custom rule that restricts the transition to the *Proposed* state category of a work item type effectively disallows creation of work items of that type. By applying the rule to a specific group, you effectively disallow that group from creating work items of that type.

Inheritance process

The following custom rule restricts a project team from creating work items as the *Proposed* state category maps to the *New* workflow state.



Restrict modification of work items by a group

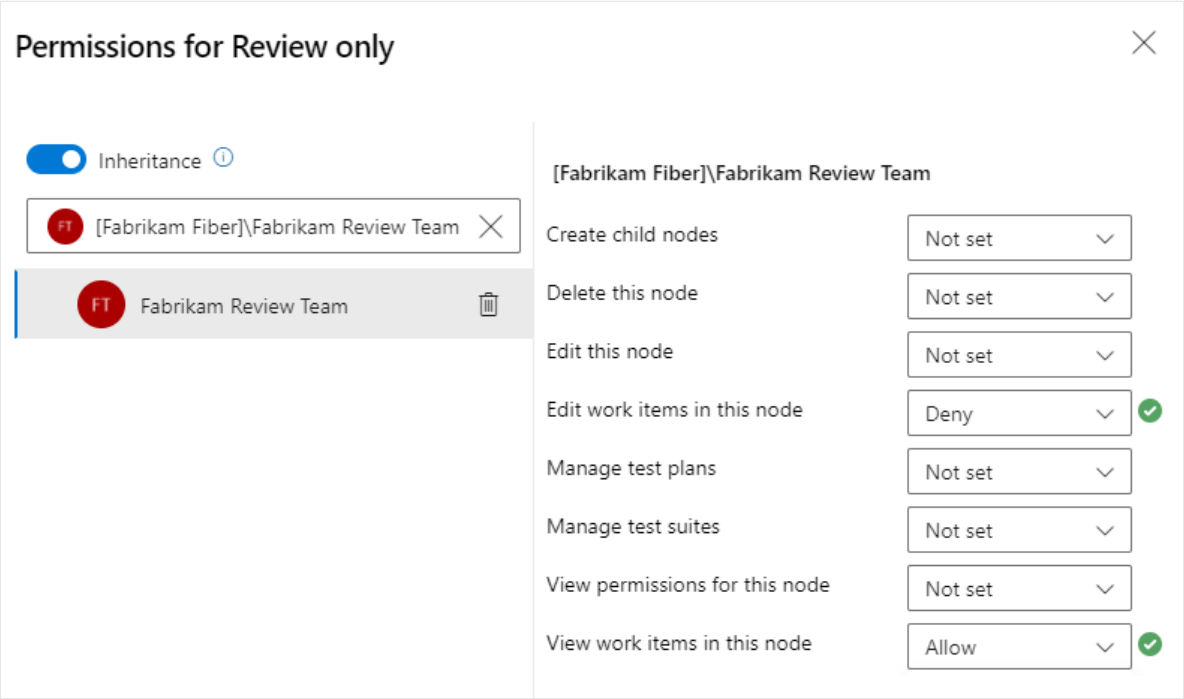
For an Inheritance process, you can prevent users from modifying a work item by setting the deny permission for a group on an Area Path. For an On-premises XML process, you can place restrictions on each workflow state for a group that prevents them from saving the work item in any state.

Inheritance process

It's not possible to define a custom rule that restricts modification of work items of a specific type. You can only specify restriction by state. If the user doesn't change the state, then they can modify other fields, unless all fields are made read-only for the group.

Instead, if you want to restrict a group of users from modifying select work items of any type, you can assign those work items to an Area Path. Define a security group,

and then set restrictions for editing work items for that Area Path for that group as shown in the following image. To learn more, see Set permissions and access for work tracking, Create child nodes and modify work items under an area path



Restrict state transitions

For inherited processes, any-to-any state transitions are automatically defined. This allows users to advanced the workflow state from new to completed, but also to move backwards in case that is needed. When defining custom rules to restrict a transition, keep in mind that if a user makes a mistake in updating the workflow, they may not be able to correct it. For example, they could update the status by moving a work item card to a later stage on the Kanban board, but not move it back.

> **Tip**
>
> Consider restricting a state transition for some but not all users. That way, if a user makes a mistake, they can ask another team member to reset the State value to bypass the restriction.

Before defining state transition rules, review Rules and rule evaluation, Auto-generated rules and How workflow states and state categories are used in Backlogs and Boards.

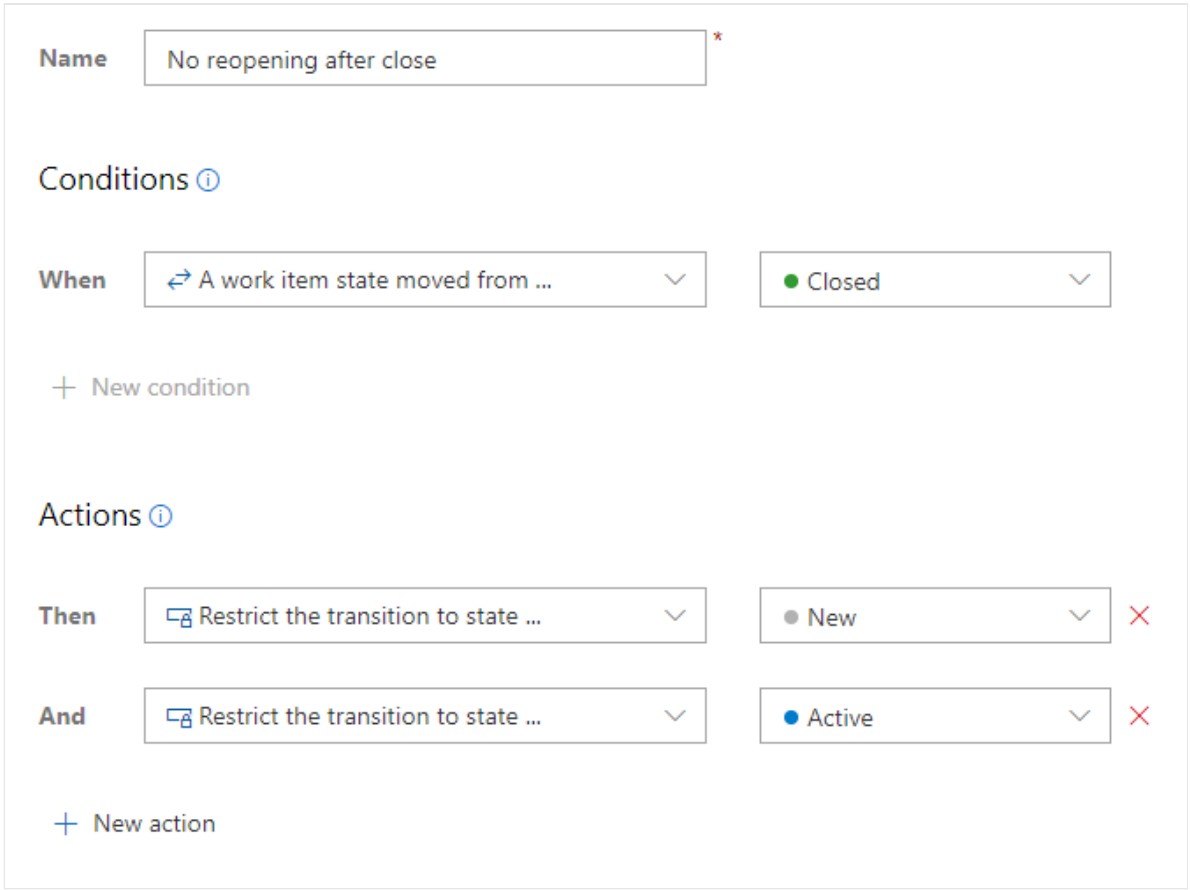
Restrict modification of closed work items

Depending on your business processes, you may want to prevent users from continuing to modify or update work items that have been closed or completed. You can add rules to work item types to prevent users from re-opening closed work items.

Inheritance process

For the Inherited process, you can add a rule that restricts state transition. For example, the following rule restricts transitioning from closed to the other two States, New and Active.

> **Note**
>
> The `A work item state moved from ...` condition is available for Azure DevOps Server 2020 and later versions.

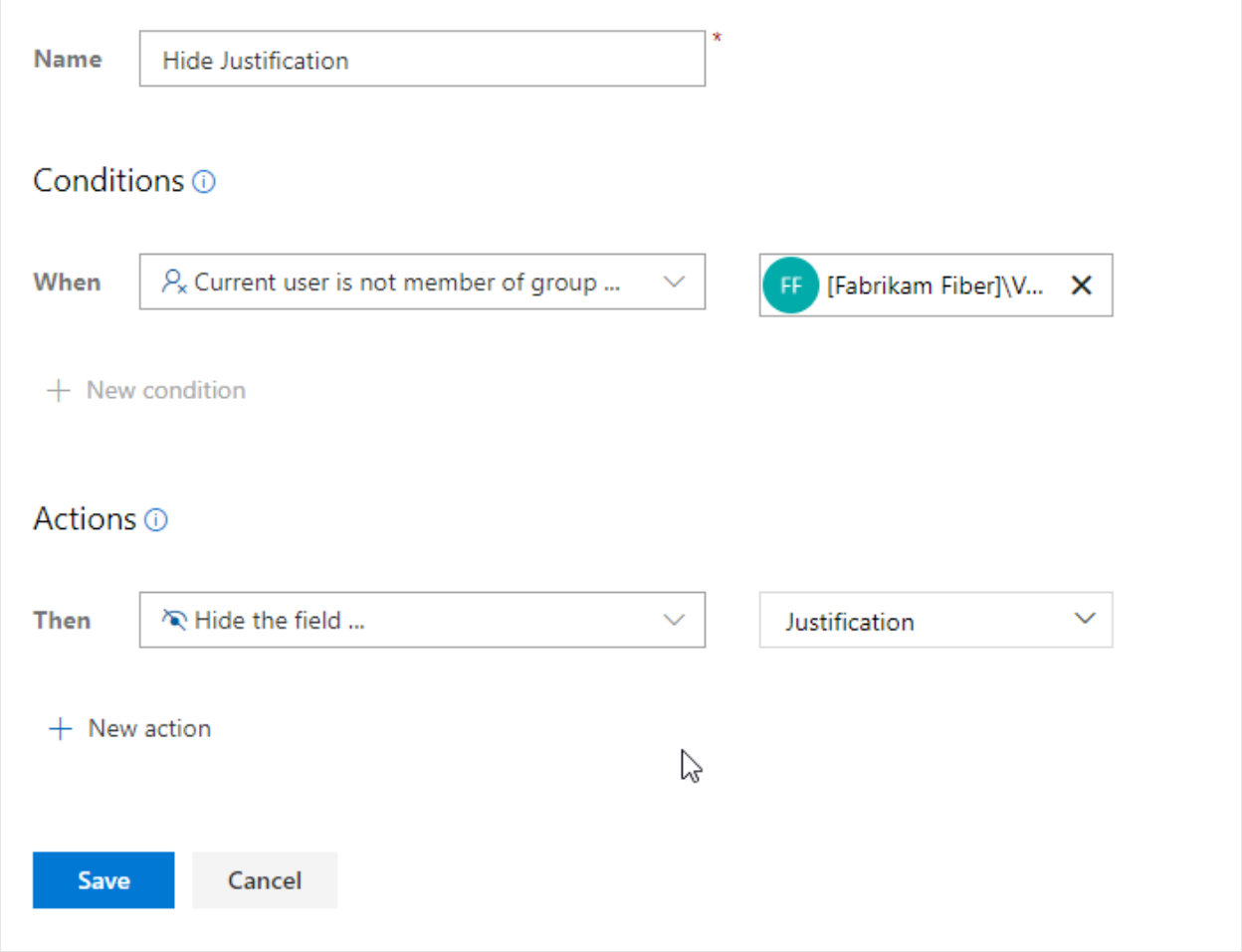


> **Note**
>
> Depending on the rule action you specify, either the **Save** button on the work item form may be disabled, or an error message displays when a restricted user attempts to modify the work item.

Hide or restrict modification of a field based on a user or group

When you select the `Current user is a member of group...` or `Current user is not a member of group...`, you can hide a field, make a field read-only, or make a field required.

For example, the following condition indicates that the Justification field is hidden for members who don't belong to the Fabrikam Fiber\Voice group.



> **Note**
>
> Work items are subject to rules applied to them. Conditional rules based on user or group membership are cached for your web browser. If you find yourself restricted to update a work item, you may have encountered one of these rules. If you believe you've encountered an issue that doesn't apply to you, see Work item form IndexDB caching issues.

Restrict modification of select fields based on a user or group

You can customize work item types to restrict who can modify a specific field for a work item type.

Inheritance process

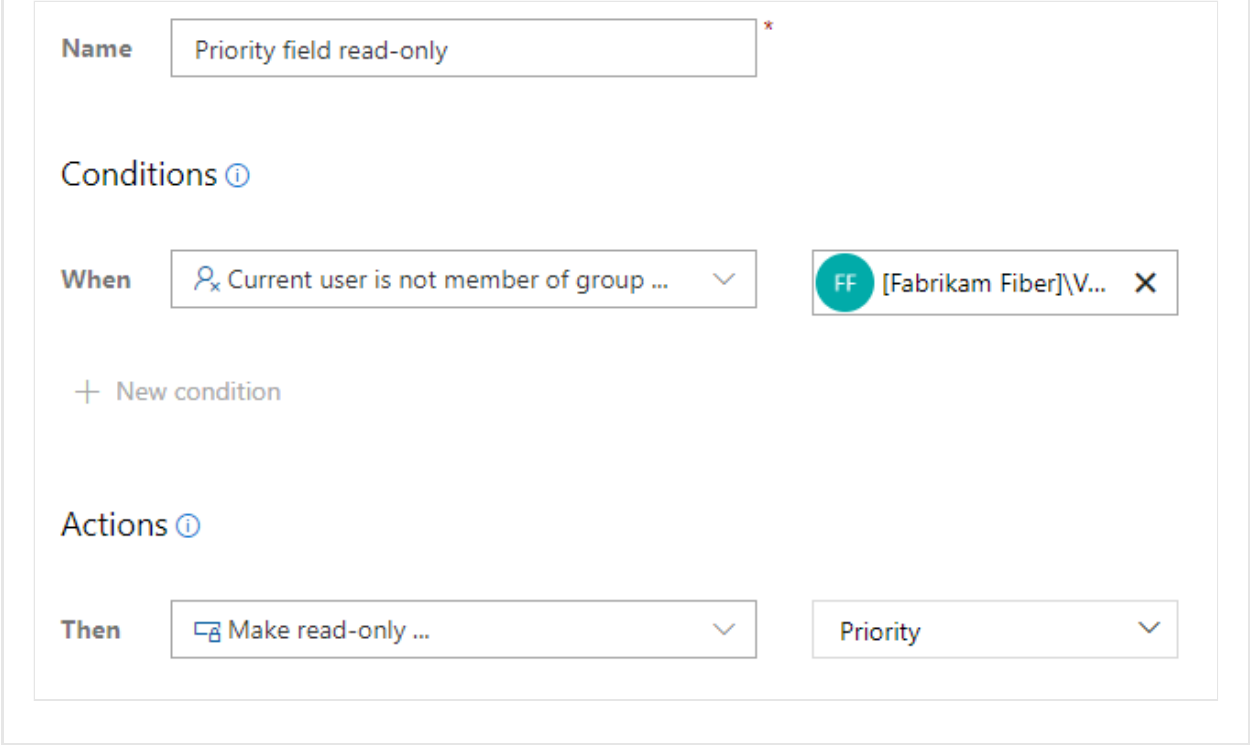
Using one of the following two conditions, you can make select fields required for a user of a security group or who are not a member of a security group.

- `current user is a member of a group...`
- `current user is not a member of a group...`

> **Tip**
>
> To avoid rule evaluation issues that may arise, specify Azure DevOps security groups and not Microsoft Entra ID or Active Directory security groups. To learn more, see **Default rules and the rule engine**.

For example, you can make the **Title** or the **State** fields **Read-only** for select users or groups.

For example, the **Priority** field, for the User Story work item type, becomes read-only for members of the Fabrikam Fiber\Voice group. When a user of this group opens a User Story, they are unable to change the value on the Priority field.



Related articles

- Apply rules to workflow states (Inheritance process)
- Rules and rule evaluation

Feedback

Was this page helpful? Yes No

Provide product feedback

Change a project process from Agile to Scrum

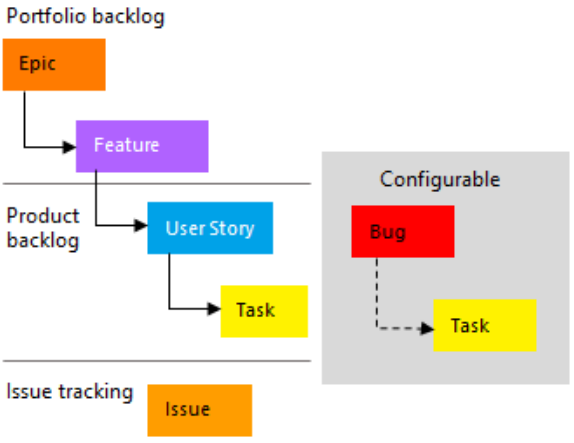
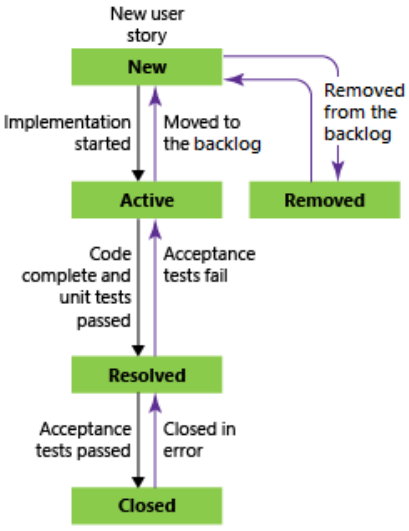
Article • 07/24/2024

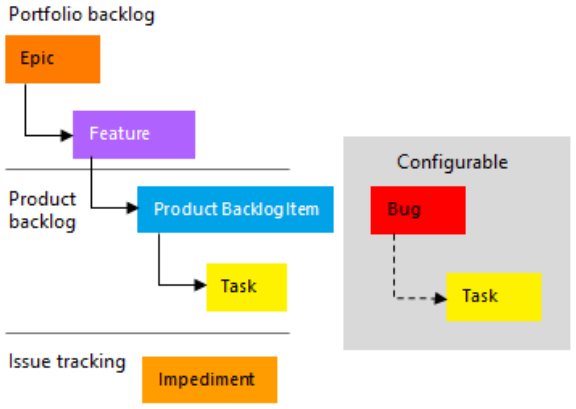
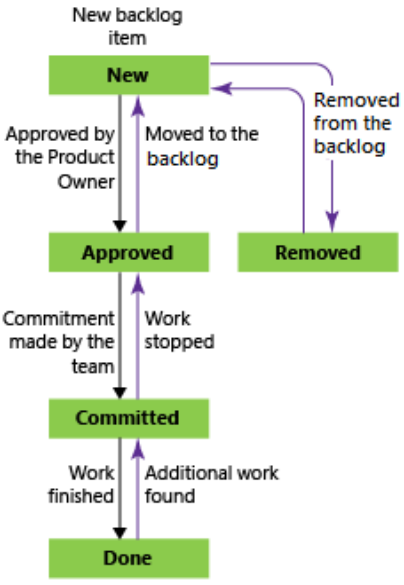
Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

You can change a project based on the Agile process to use an inherited Scrum process. This article provides the steps needed to make this change.

Prior to making this change, we recommend you familiarize yourself with the process you are changing to. The Task, Bug, Feature, and Epic work item types are the same for both Scrum and Agile processes. Some State and Reason field values, however, are different.

Expand table

Process	Work item types	Workflow
Agile		

Process	Work item types	Workflow
Scrum		

For more information, see About processes and process templates.

Reasons you might want to change your process from Agile to Scrum:

- You want to use Product Backlog Item and Effort instead of User Story and Story Points.
- Your organization is requiring everyone to standardize their tracking with a customized inherited process based on the Scrum process.

> ⚠ **Warning**
>
> If you have existing work items, this process requires manual updates to work items and board configuration. Make sure you follow the steps provided in this article to ensure you address the manual steps required after you change the process used by a project.

Prerequisites

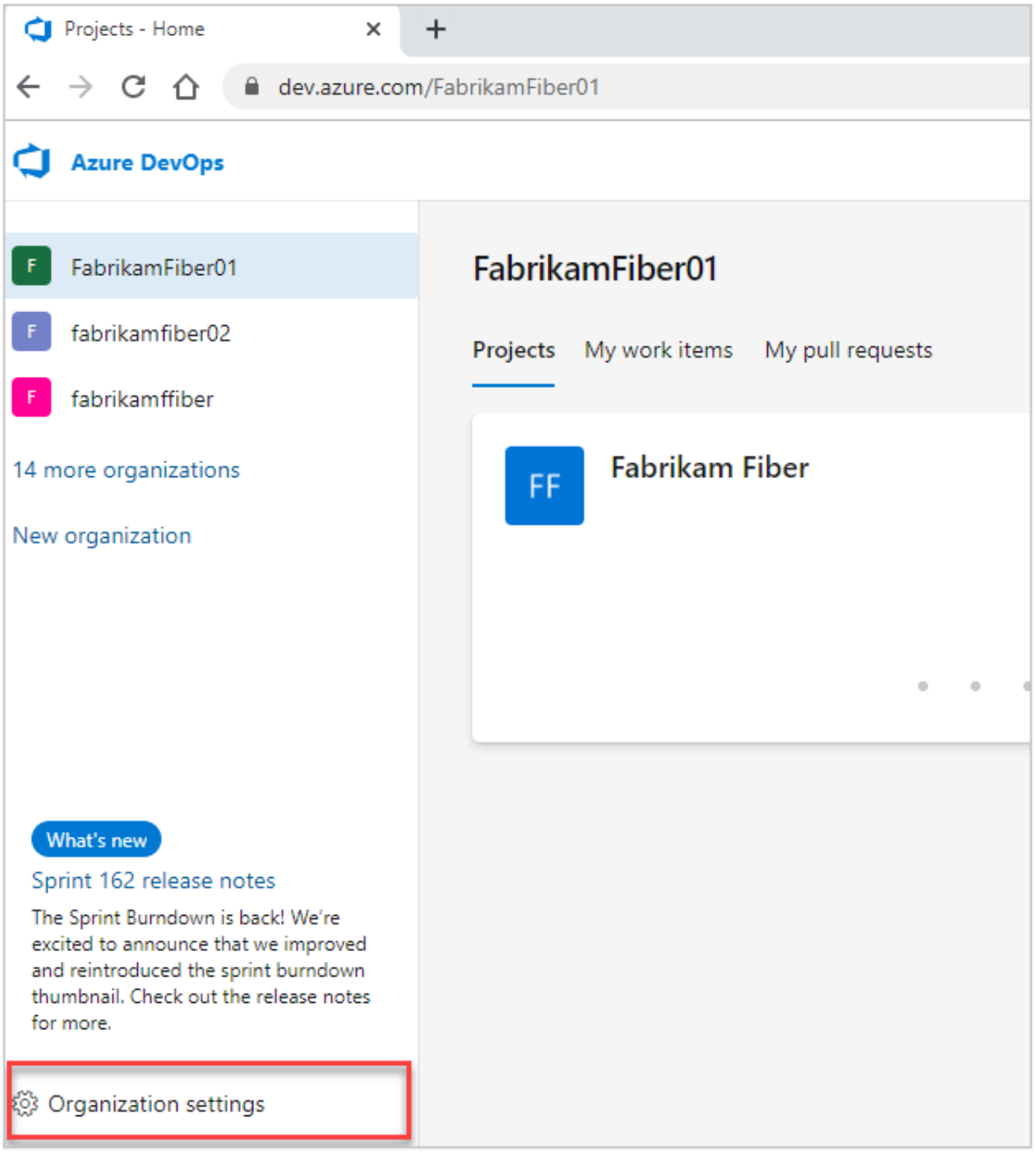
- **Permissions:**
  - To create, delete, or edit a process, be a member of the **Project Collection Administrators** group, or have specific collection-level permissions **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization**. For more information, see Set permissions and access for work tracking, Customize an inherited process.
  - To update boards, be the Team Administrator or a member of the **Project Administrators** group.

- **Access:**
  - Even if you have **Basic** or **Stakeholder** access, you can still change a process if someone gives you permissions to do so.
  - To update and change the type of your existing work items, be a member of the project.

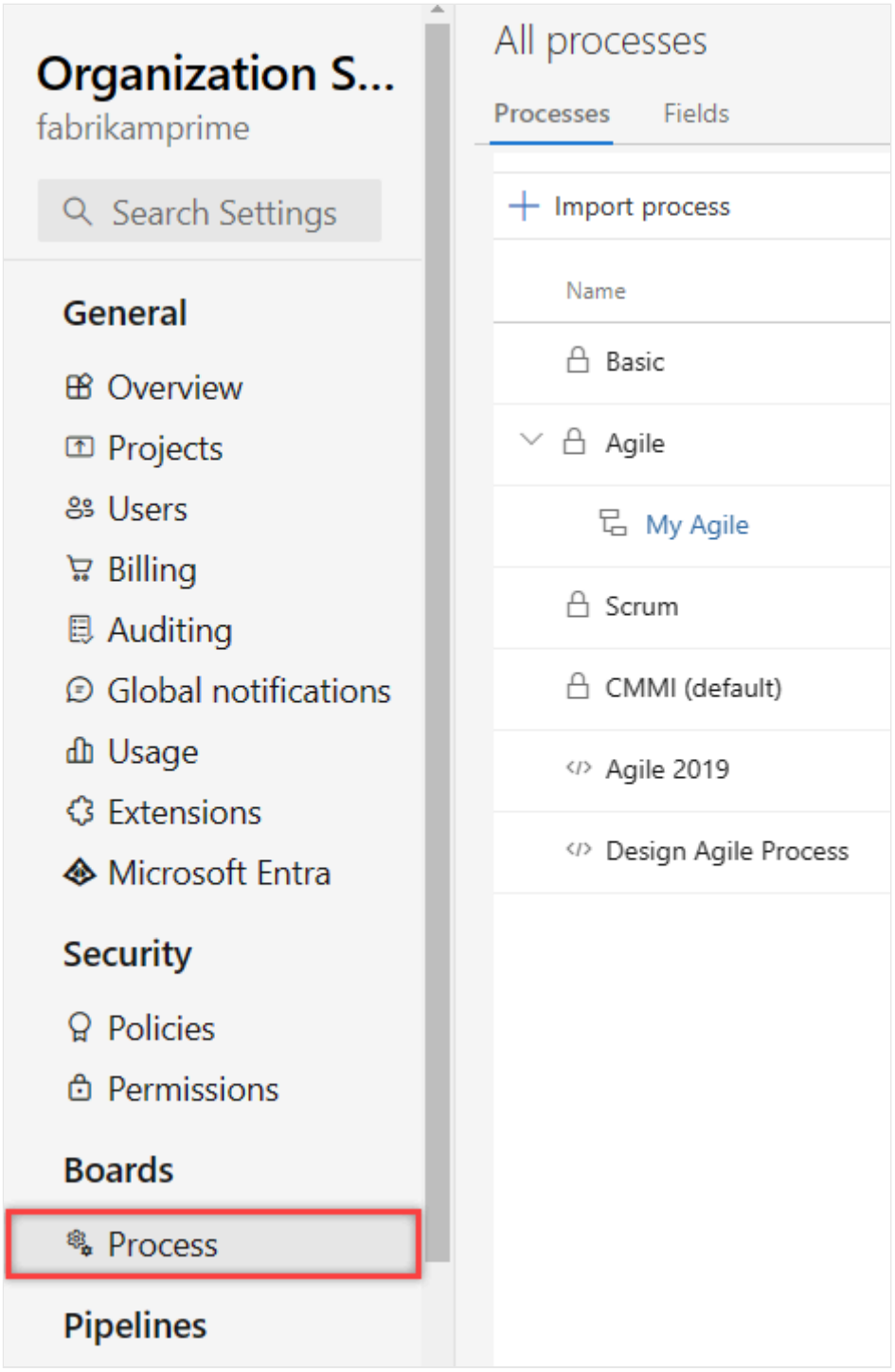
Open Settings>Process

You create, manage, and make customizations to processes from **Organization settings>Process**.

1. Choose the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.



2. Then, choose **Process**.



> **Important**
>
> If you don't see **Process**, then you're working from TFS-2018 or earlier version. The **Process** page isn't supported. You must use the features supported for the On-premises XML process model.

Create and customize a Scrum inherited process

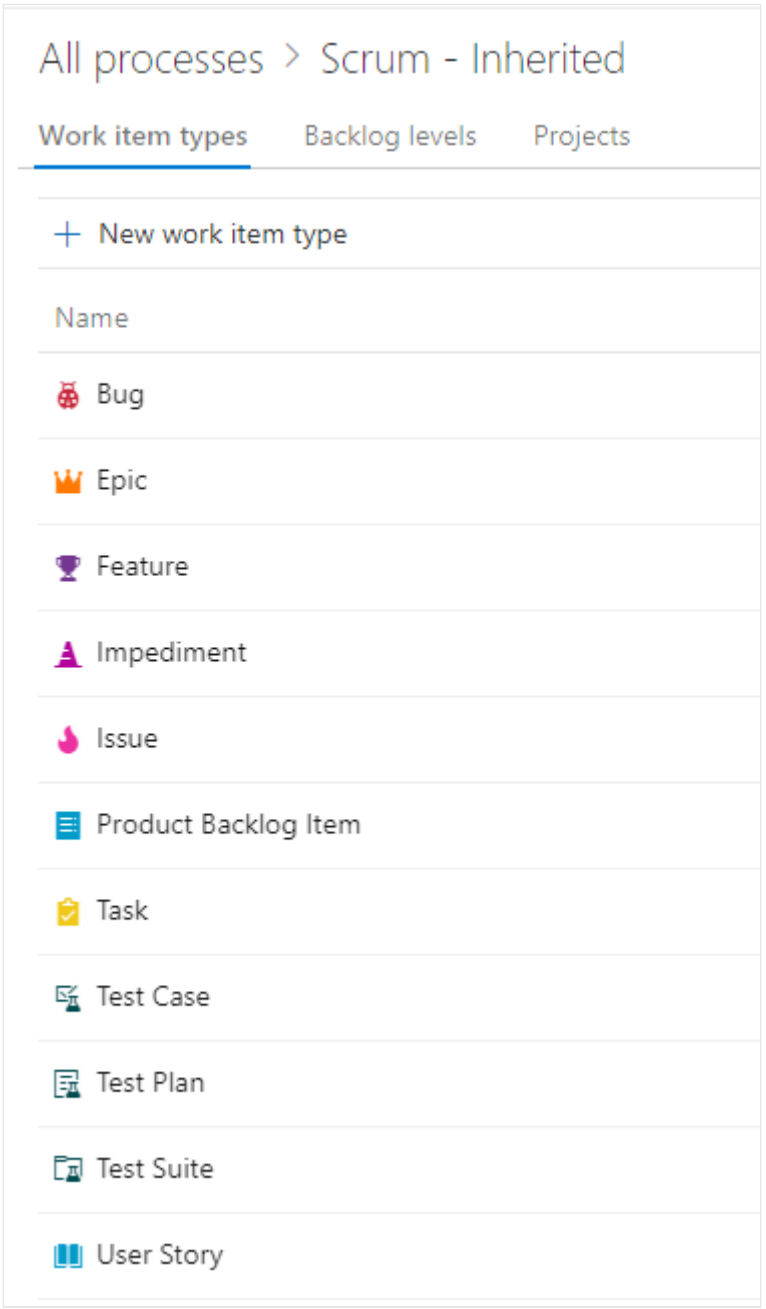
The first step is to create an inherited process that you can customize. The default, system processes are locked from being customized.

1. From the **Process** page, choose the ··· actions icon of the Scrum system process, and then select **Create inherited process**. Name the process "Scrum - Inherited".

2. If the User Story and Issue work item types are used by your current project, then add these work item types to the "Scrum - Inherited" process.

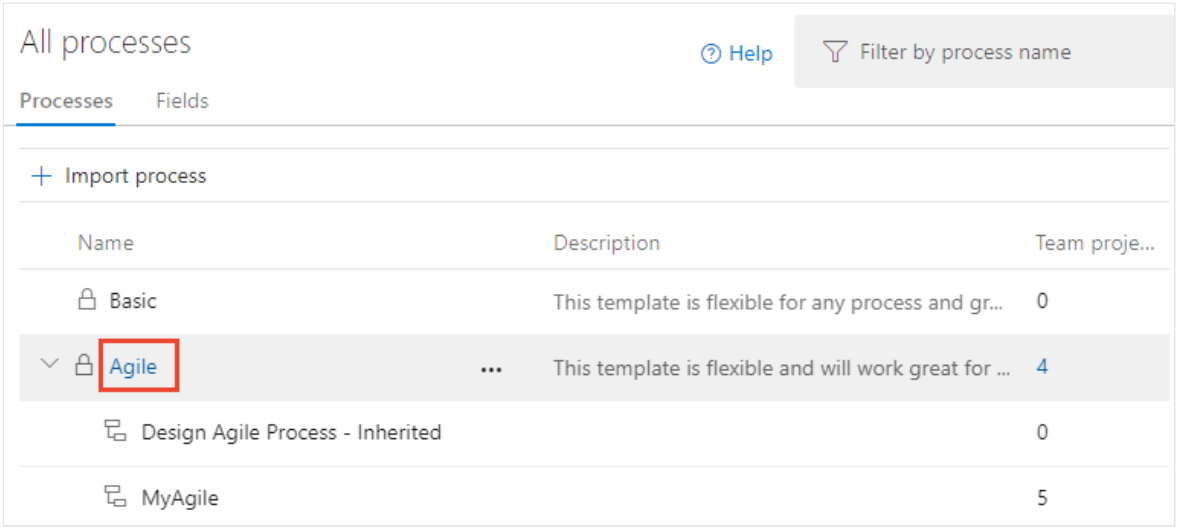
   You don't have to add any details to these work item types as you'll be changing them later to Product Backlog Item and Impediment. However, if you have captured details within a field—such as Story Points, Priority, or other field—then add those fields to the work item types.

3. When you are finished, your work item types list should appear as follows.

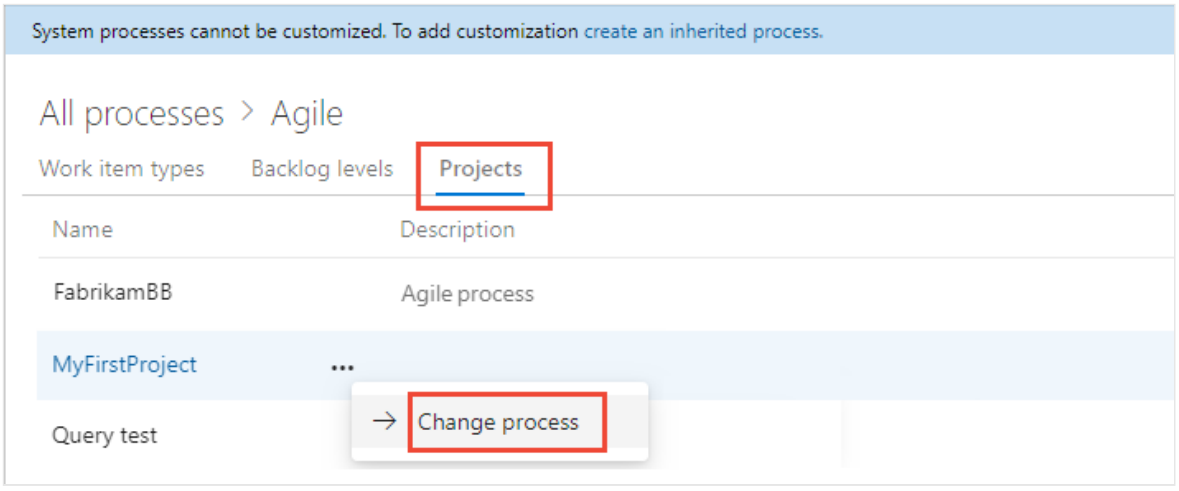


Change the project process

1. Choose the process that contains the project you want to change. To change from Agile to Scrum, choose **Agile** or the custom process based on Agile. Here we choose Agile.

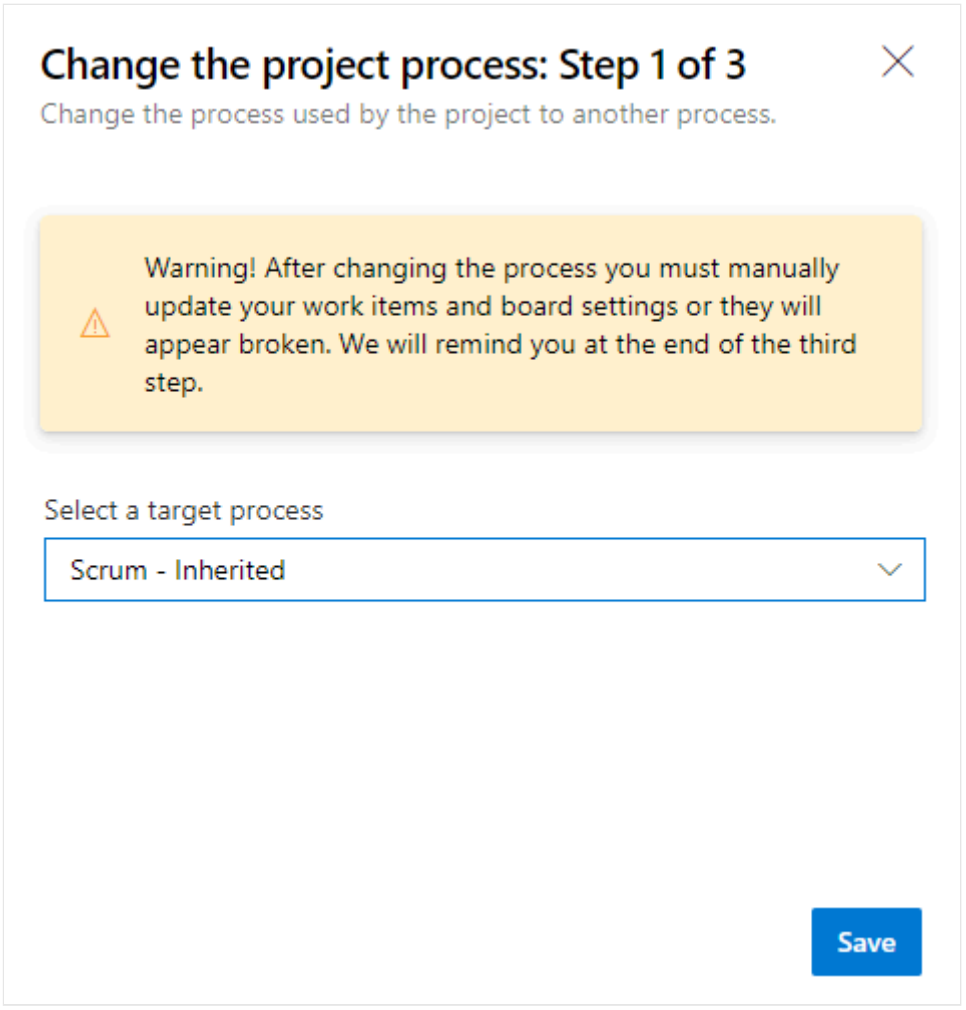


2. Choose **Projects**, and then choose the ··· actions icon for the project you want to change, and select **Change process**. Here we choose the MyAgile project.

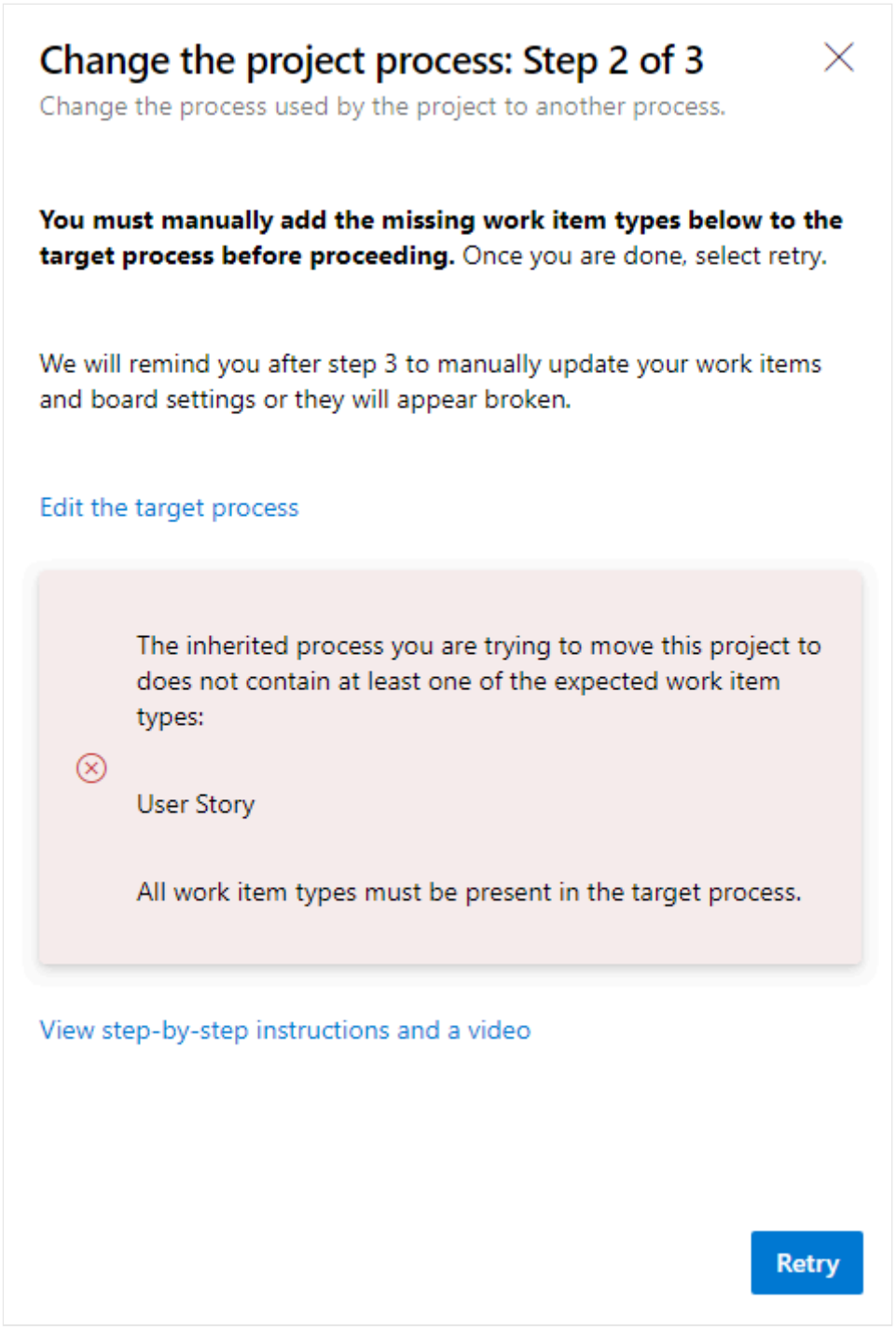


   Follow the steps in the wizard.

3. Choose the **Scrum - Inherited** process that you created earlier and then choose **Save**.

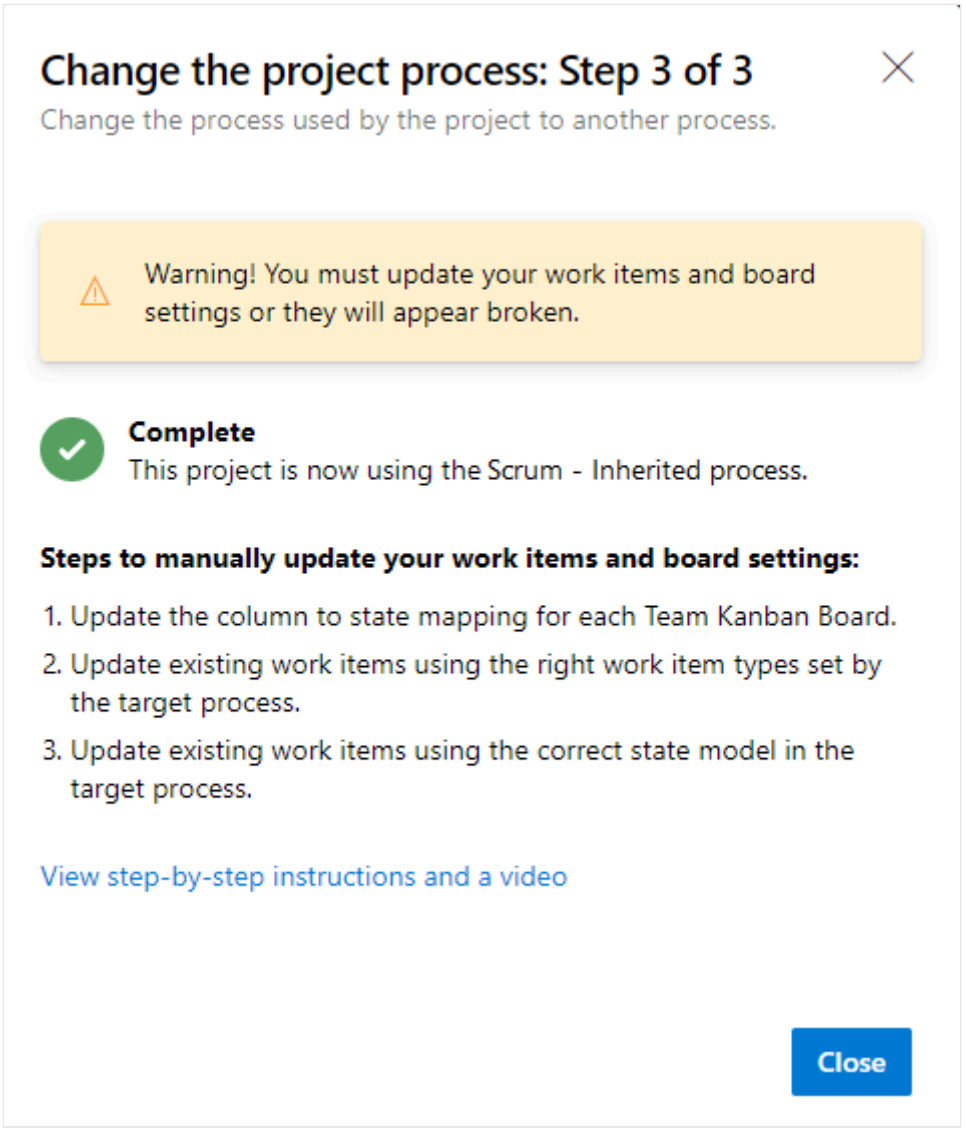


If you skipped step 2 of Create and customize a Scrum inherited process, you'll receive the following message to add the missing work item types.



Close the wizard dialog and add the missing work item types now. See Add and manage work item types.

4. Upon successful completion, the wizard displays the following information. Make a note of the steps to follow and then choose **Close**.



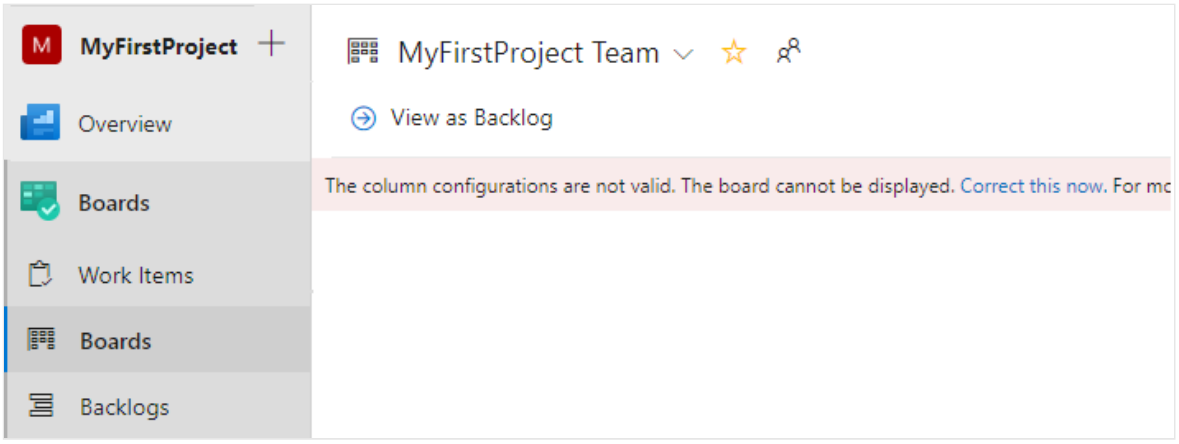
Steps to manually update your work items and board settings:

- Update the column to state mapping for each team board
- Update existing work items using the work item types set by the target process
- Update existing work items using the correct state model of the target process.

Update board column-to-state settings

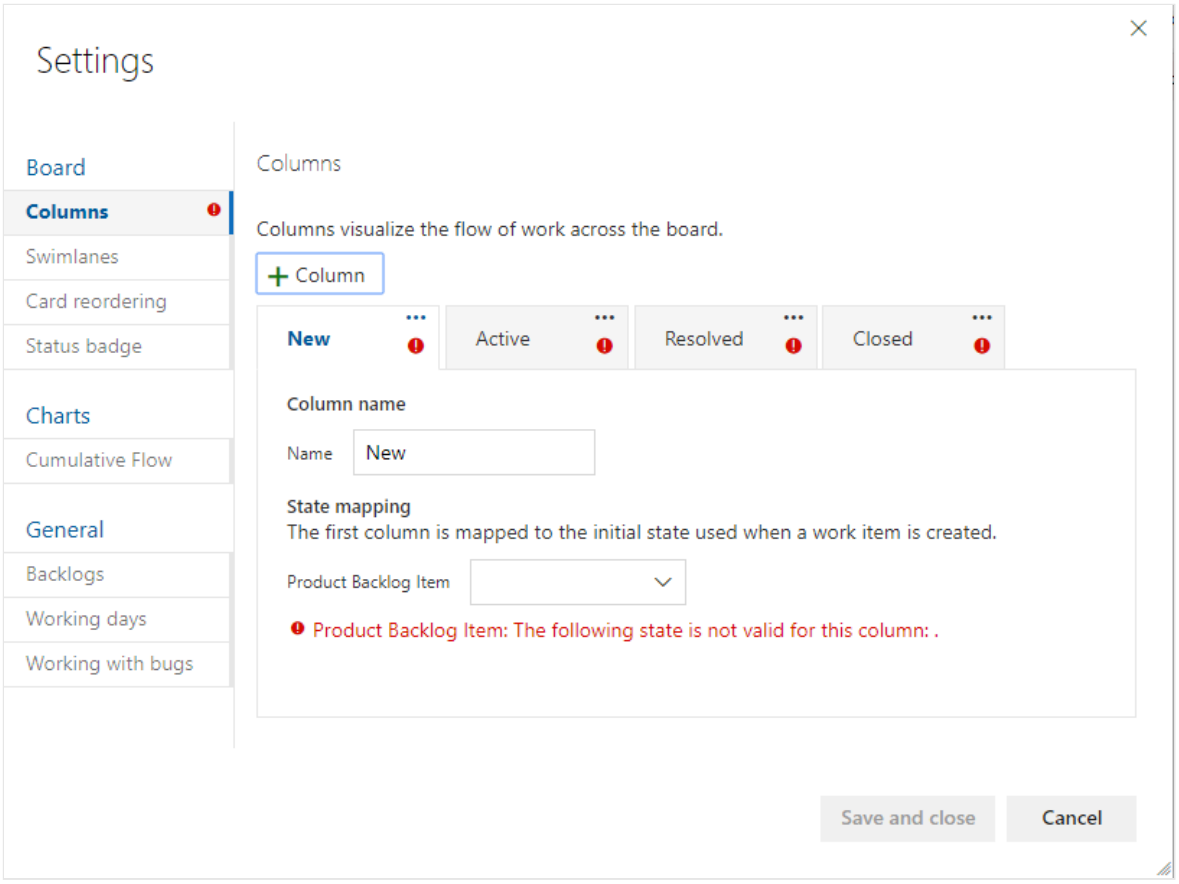
You can customize boards to display intermediate columns. For each column added, you must choose a valid workflow state for the work item types displayed on the board. For more information, see Workflow states & state categories.

1. For each team, open your board.

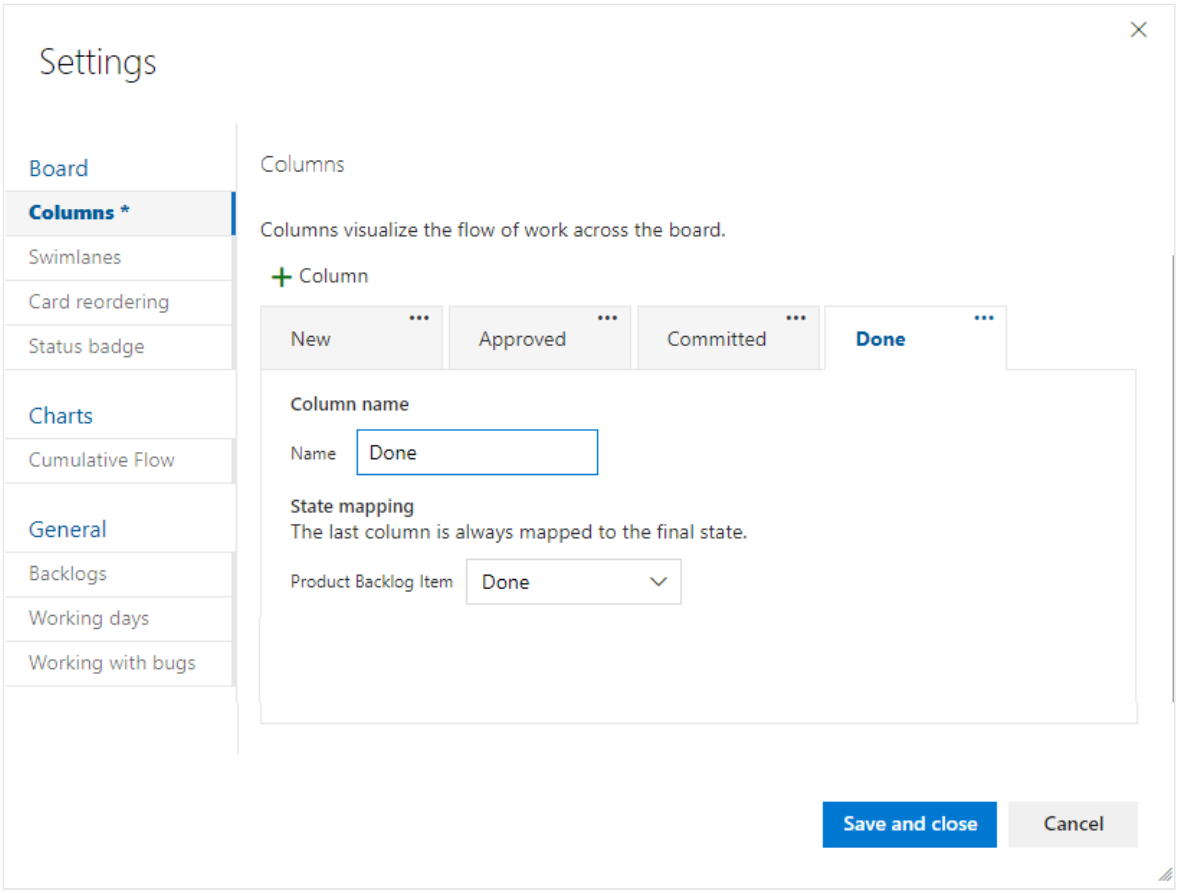


2. Choose the **Correct this now** link or the gear icon to configure the board settings.

   The Settings dialog opens. Those tabs that display a required icon need correction.



   Rename each column and choose the correct state for each column so that the column-to-state mapping is correct. As needed, add one or more columns. When done, choose **Save and close**.

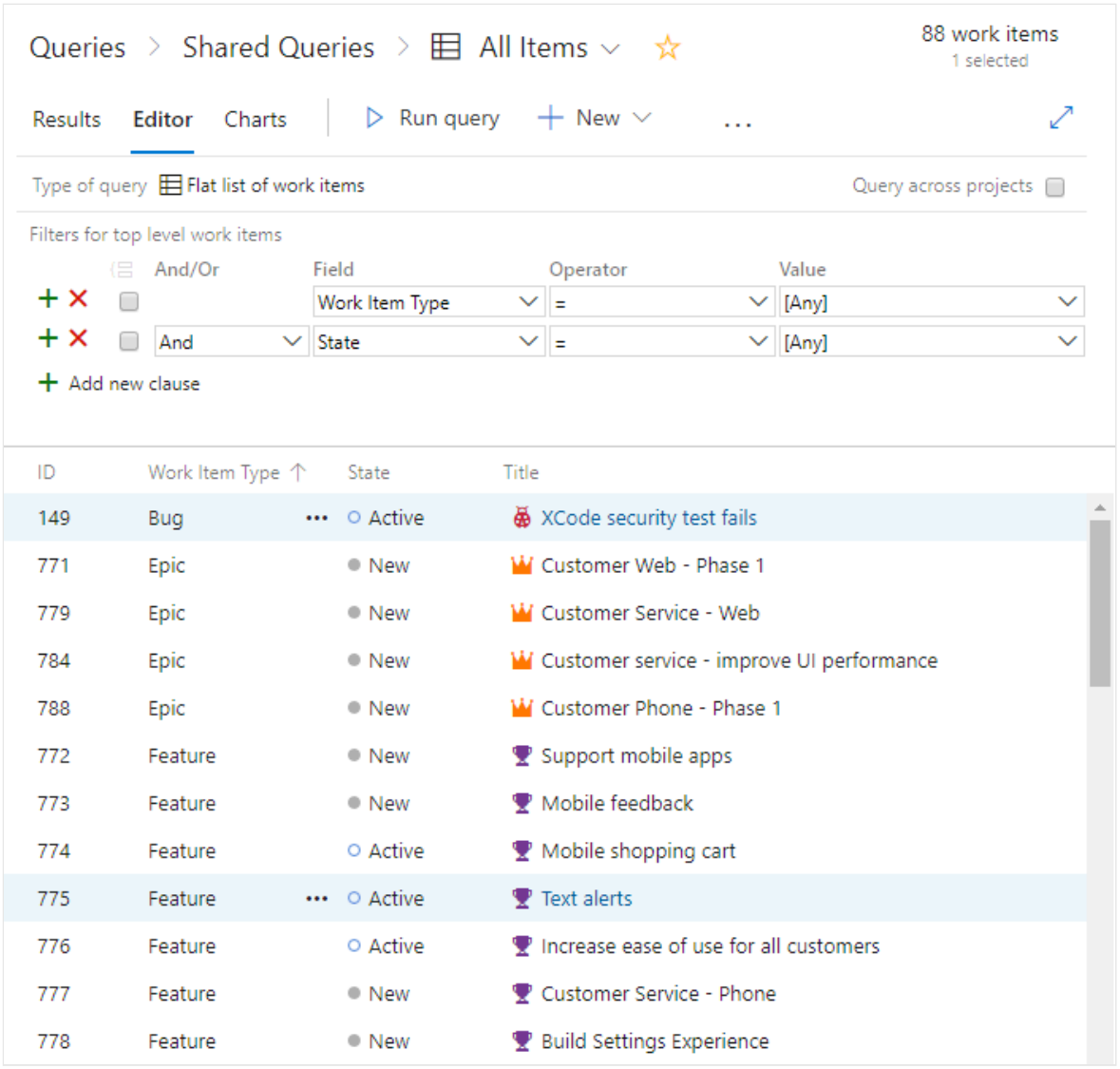


Update work items

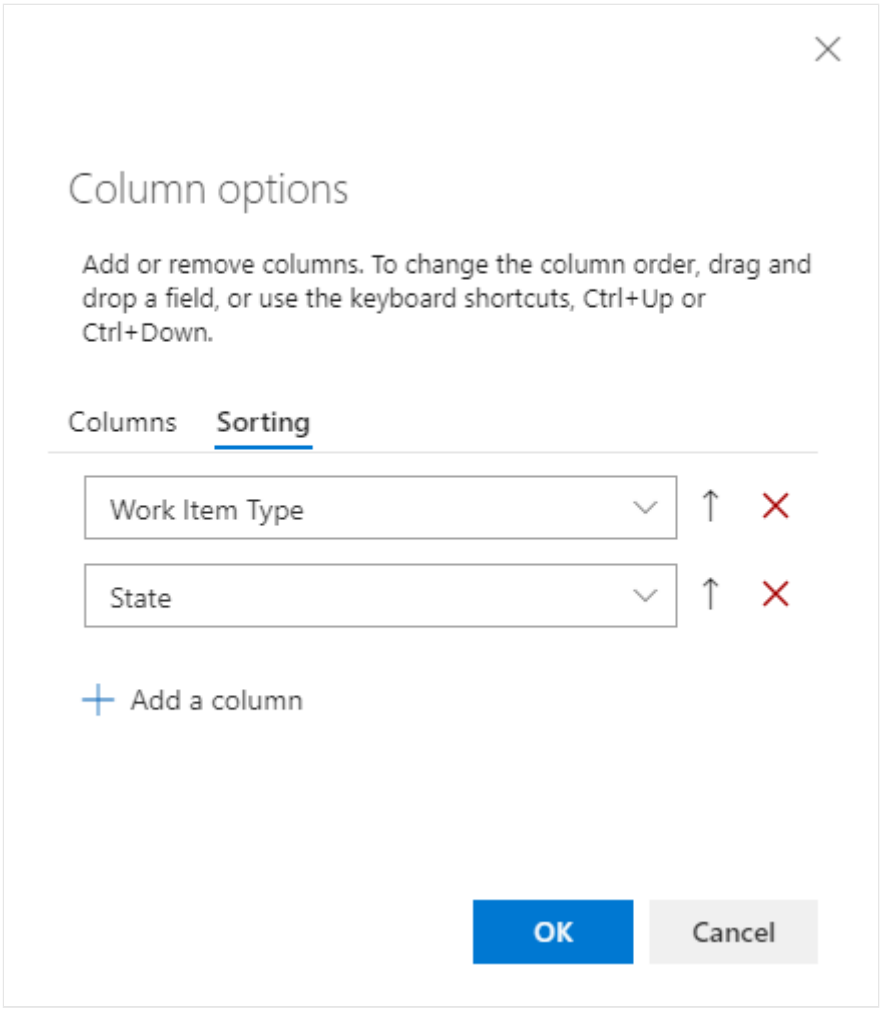
Your next step is to bulk update work items. The recommended sequence is:

- Create a work item query that displays all work items that need update.
- Perform a bulk update to change the work item type of User Story work items to Product Backlog Items.
- Perform a bulk update on all States to change from Agile states—Active, Resolved, and Closed—to Scrum process states—Approved, Committed, and Done.

1. Create a query to get a list of all User Stories, Issues, Features, Tasks, and Epics. You will need to change the work item type of the first two work item types, and the State and Reason fields for all items.

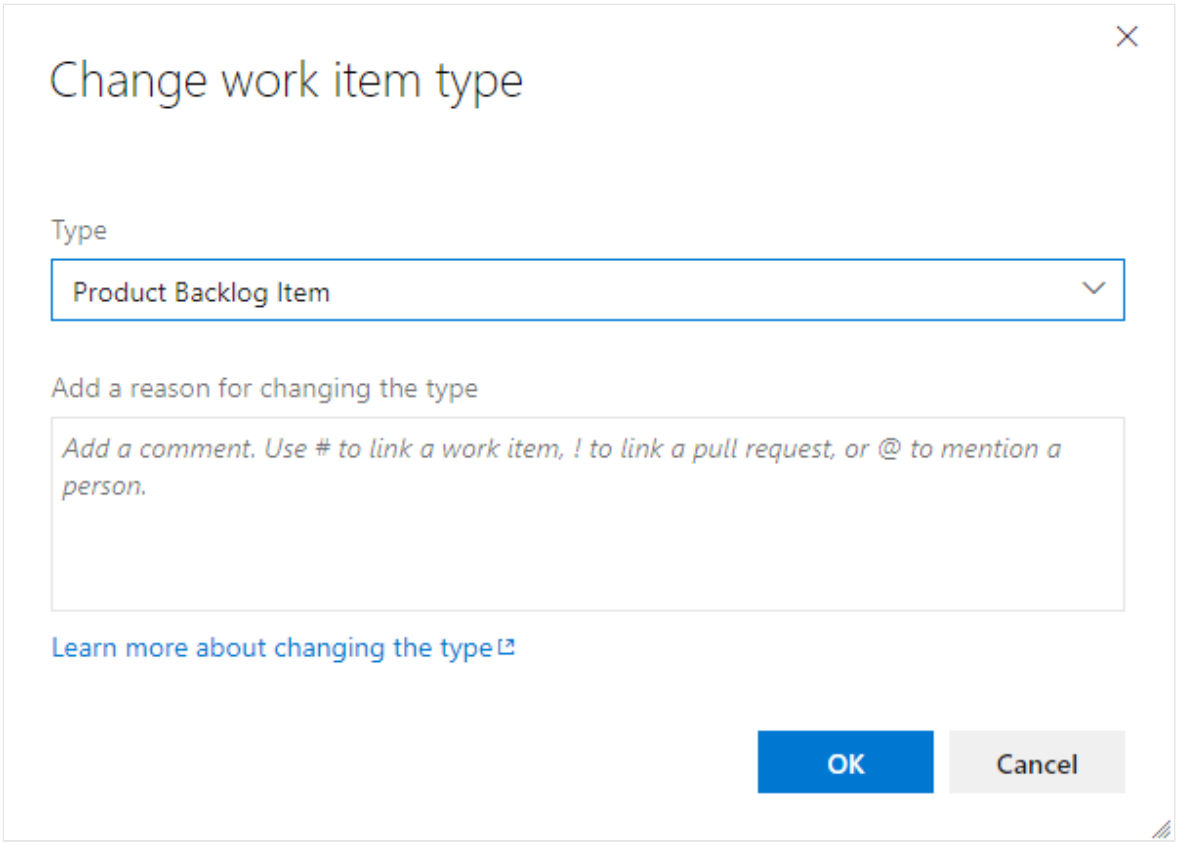


2. Choose the ··· actions icon and then select **Column options**. Choose to show the State and Reason fields. Choose the **Sort** tab, and set it to sort the list by work item type and state value.



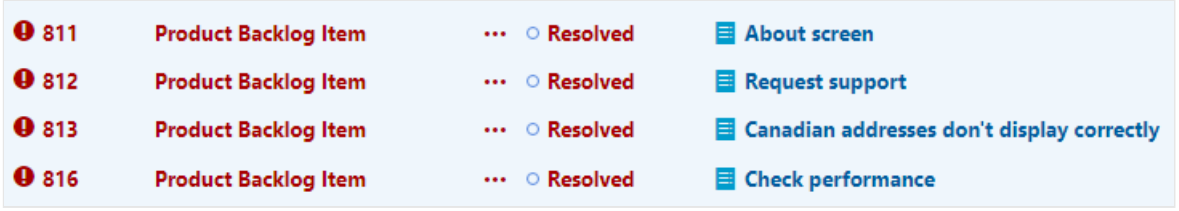
3. Choose **Results** to just show the list of work items.

4. Highlight all User Stories, choose the ··· actions icon, select **Change type**, and change the type to Product Backlog Item.

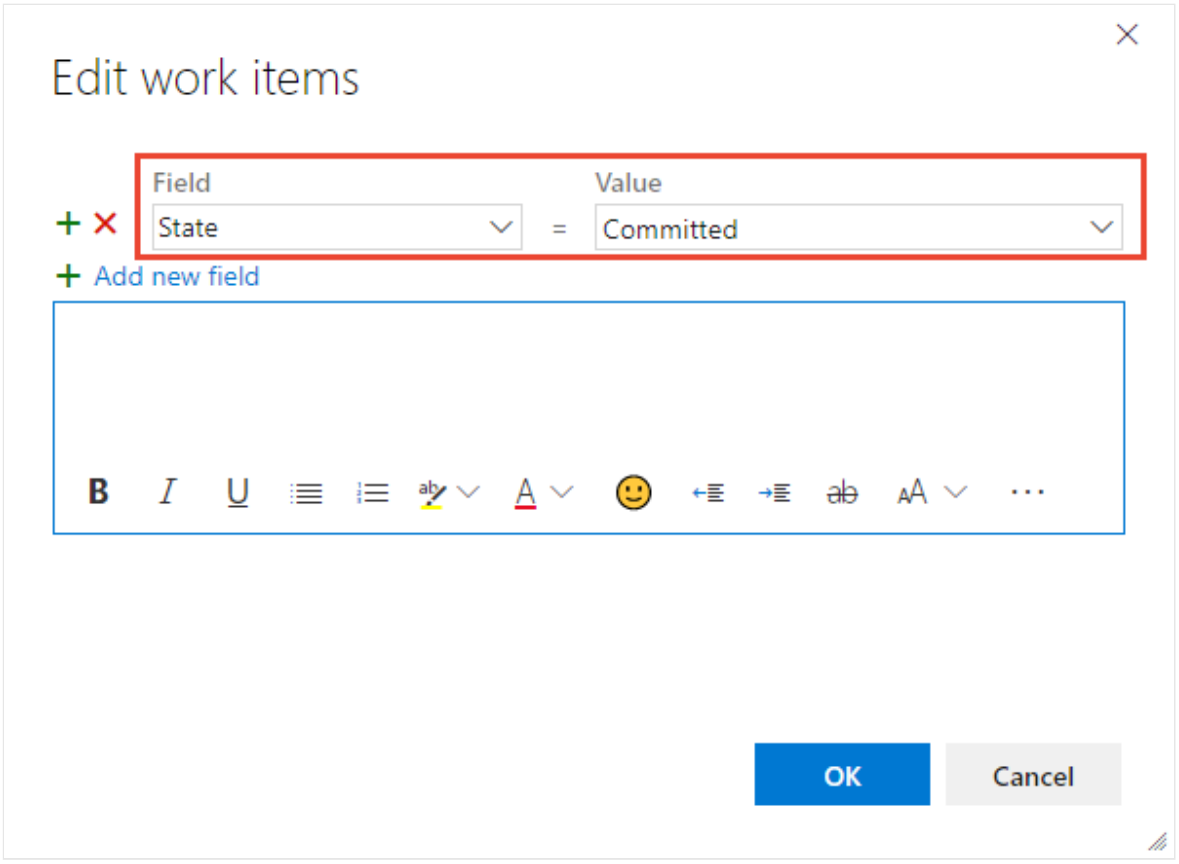


For more details, see Move, change, or delete work items, Change the work item type.

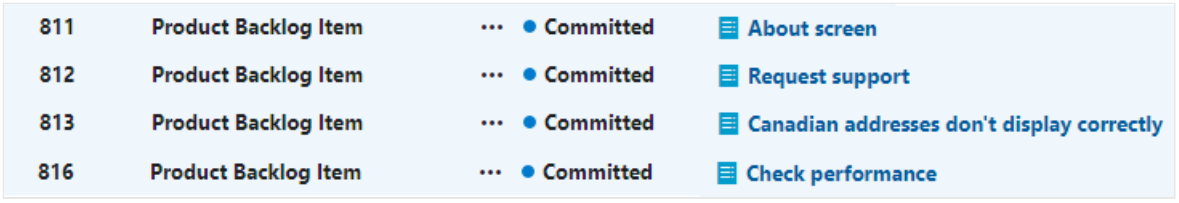
It's possible that you will receive errors where the work item type and the state are mismatched. In that case, you can't save your changes until you update the state as described in the next step.



5. Highlight all work items of the same State, such as Resolved, choose the ··· actions icon, and then select **Edit**. Add the State field and select Committed or a valid value for the Product Backlog Item. You may also need to update the Reason field. For details, see Bulk edit work items.



This should result in an updated list without errors.

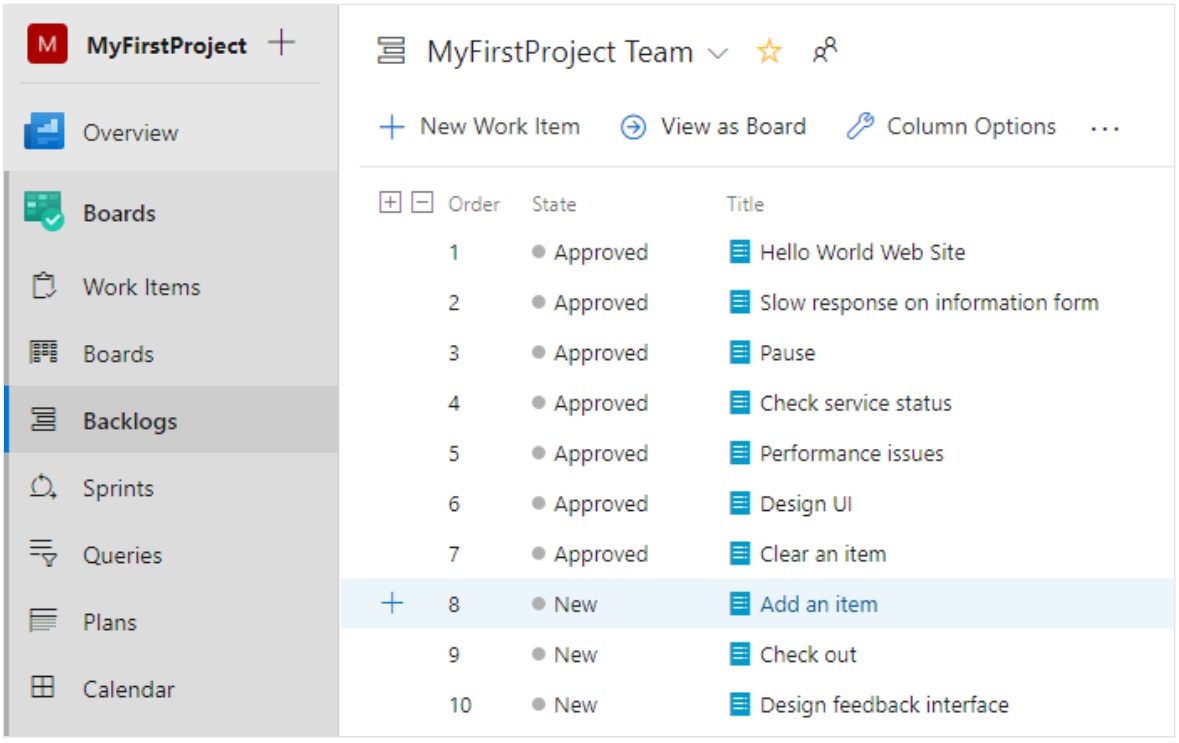


6. To save your changes, choose the ··· actions icon and select **Save items**.
7. Repeat the above steps to change the work item type for Issue to Impediment, and update the states to valid states for the work item types.
8. When done, make sure you choose the ··· actions icon and select **Save items**.

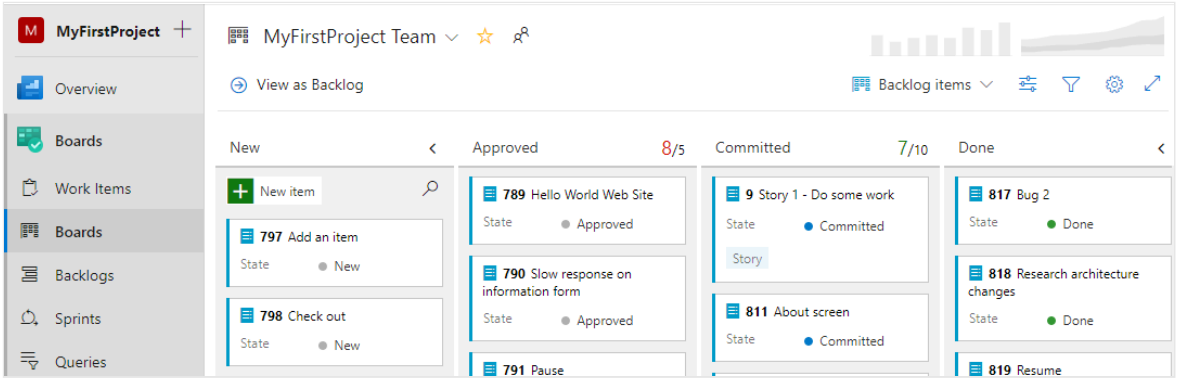
> **Tip**
>
> If you have a number of field value changes, such as populating the Effort field with the value held in Story Points, you can do that using Excel. See Bulk add or modify work items with Excel.

Verify your changes

1. Go to your team backlog and review the user stories.



2. Go to your team board and verify that the column settings are valid.



To add columns or change column names, see Add columns to your board.

Optional updates

After changing the process, you may want to make additional updates as follows:

- Change the project name
- Update your project summary or vision
- Disable or delete the work item types that won't be used going forward. See Customize a work item type for an inherited process

Related articles

- Bulk modify work items
- Change the process used by a project

- Create an inherited process
- Add and manage work item types
- Show bugs on backlogs and boards
- Customize your boards
- Create and saved managed queries with the query editor

Feedback

Was this page helpful? Yes No

Provide product feedback

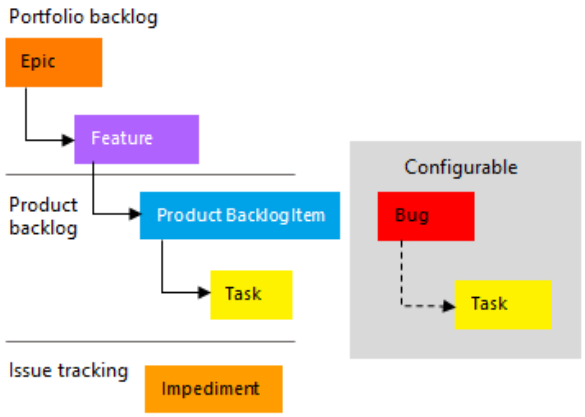
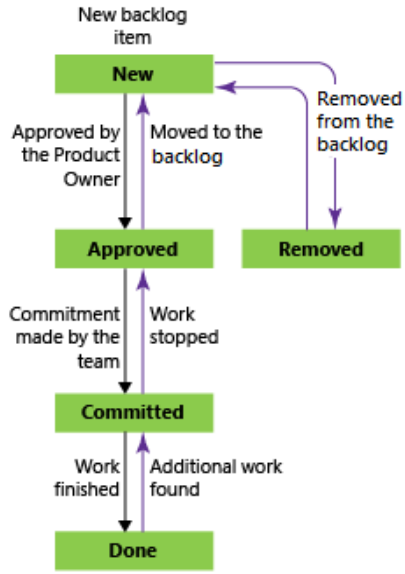
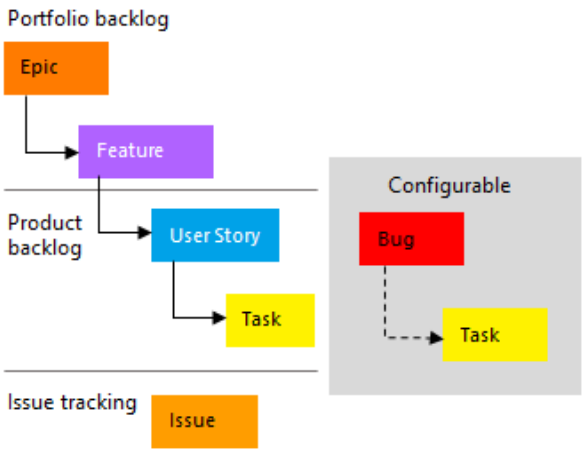
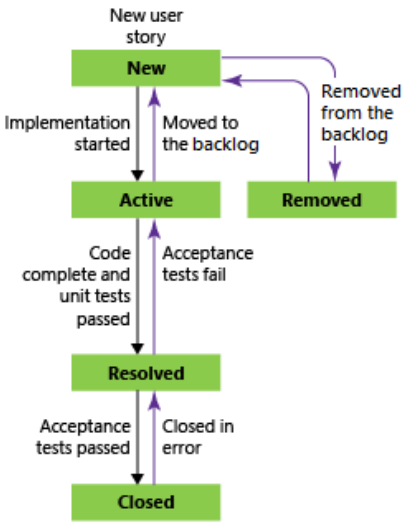
Change a project process from Scrum to Agile

Article • 03/22/2023

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

You can change a project based on the Scrum process to use an inherited Agile process. This article provides the steps needed to make this change.

Prior to making this change, we recommend you familiarize yourself with the process you are changing to. The Task, Bug, Feature, and Epic work item types are the same for both Scrum and Agile processes. Some State and Reason field values, however, are different.

Process	Work item types	Workflow
Scrum		
Agile		

For more information, see About processes and process templates.

Reasons you might want to change your process from Scrum to Agile:

- You want to use User Story and Story Points instead of Product Backlog Item and Effort.
- Your organization is requiring everyone to standardize their tracking with a customized inherited process based on the Agile process.

> **Important**
>
> If you have existing work items, this process requires manual updates to work items and board configuration. Make sure you follow the steps provided in this article to ensure you address the manual steps required after you change the process used by a project.

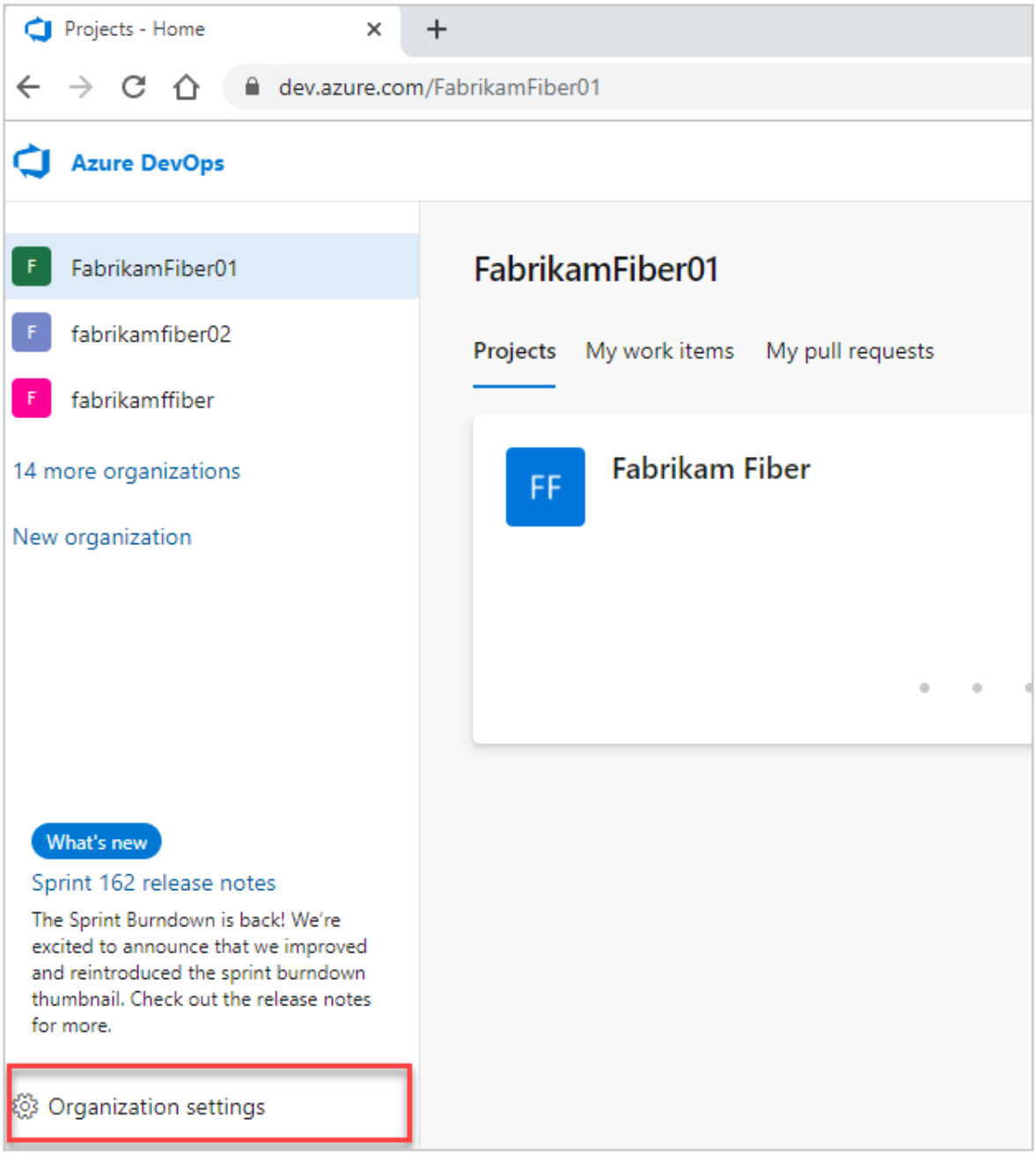
Prerequisites

- To change a process:
  - To create, delete or edit a process, you must be a member of the **Project Collection Administrators** group, or have the corresponding collection-level permissions **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization** set to **Allow**. See Set permissions and access for work tracking, Customize an inherited process.
  - Users granted **Basic** or **Stakeholder** access can be granted permissions to create, edit, and delete processes, and administer process permissions.
- To update Kanban boards, you must be the team administrator for the boards or a member of the **Project Administrators** group
- To update and change the work item type of existing work items, you must be a member of the project.

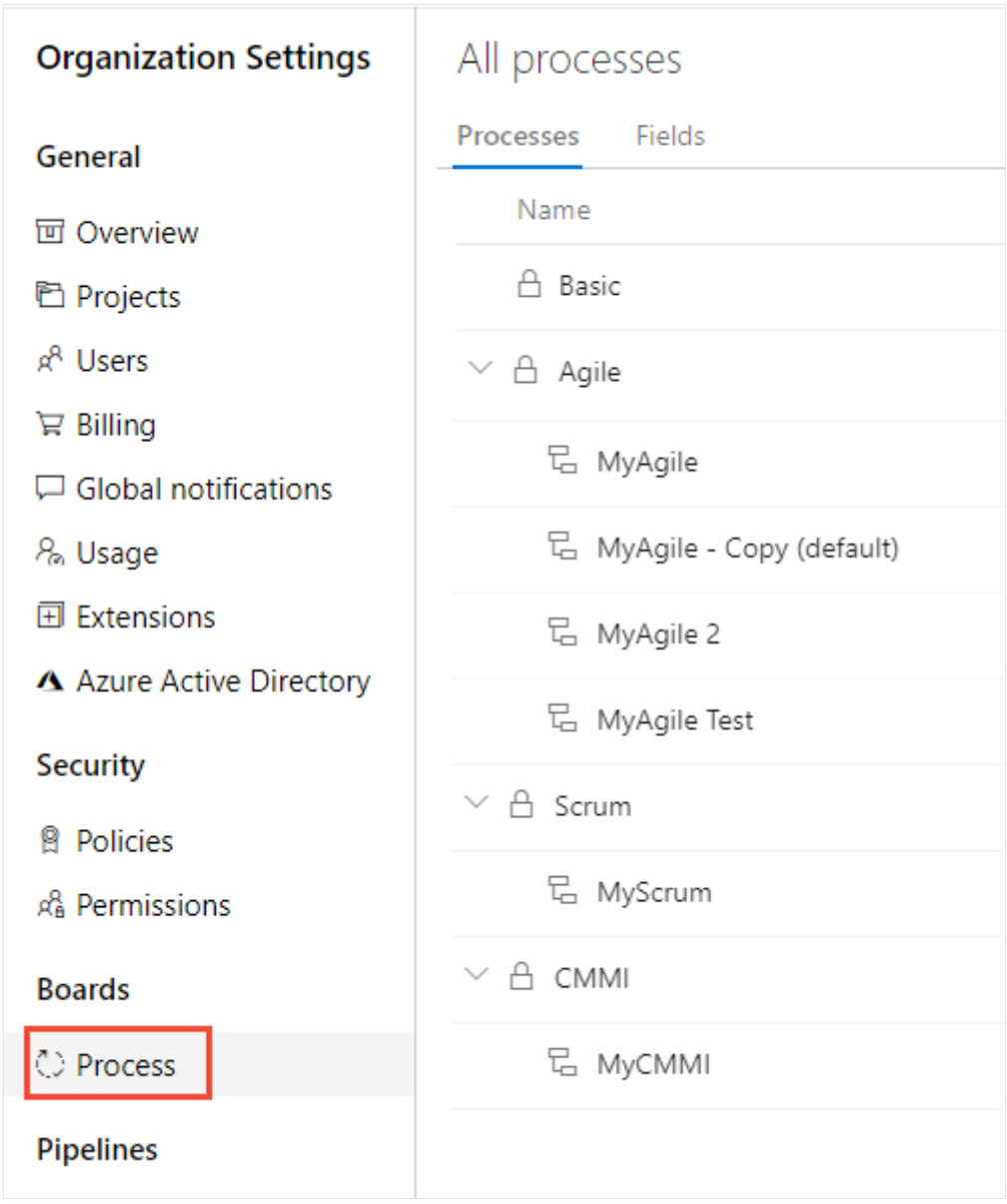
Open Settings>Process

You create, manage, and make customizations to processes from **Organization settings>Process**.

1. Choose the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.



2. Then, choose **Process**.



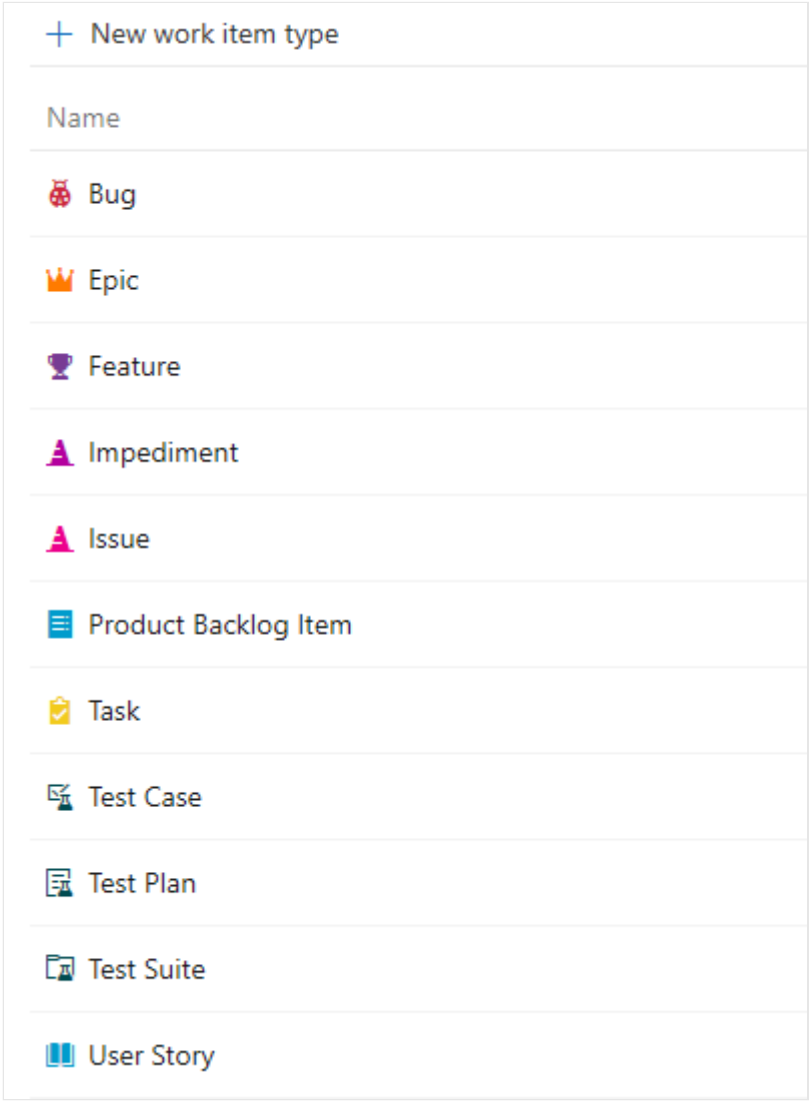
> **Important**
>
> If you don't see **Process**, then you're working from TFS-2018 or earlier version. The **Process** page isn't supported. You must use the features supported for the **On-premises XML process model**.

Create and customize an Agile inherited process

1. Create a custom process under Agile called "Agile - Inherited".
2. If the Product Backlog Item and Impediment work item types are used in your current Scrum-based project, you need to add these work item types to the "Agile - Inherited" process.

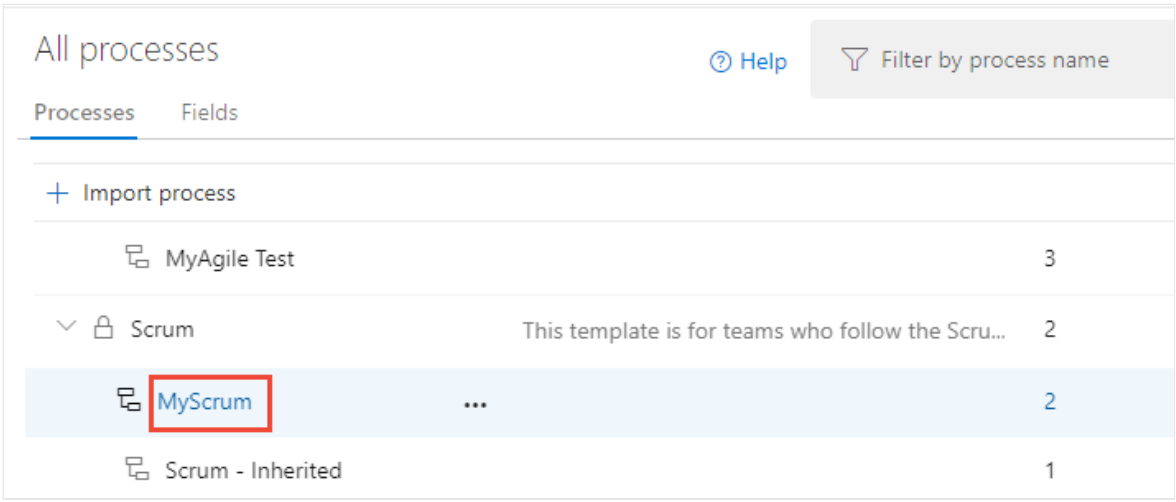
You don't have to add any details to these work item types as you'll be changing them later to User Story and Issue. However, if you have captured details within a field—such as Effort, Priority, or other field—then add those fields to the work item types.

3. When you are finished, your work item types list should appear as follows.

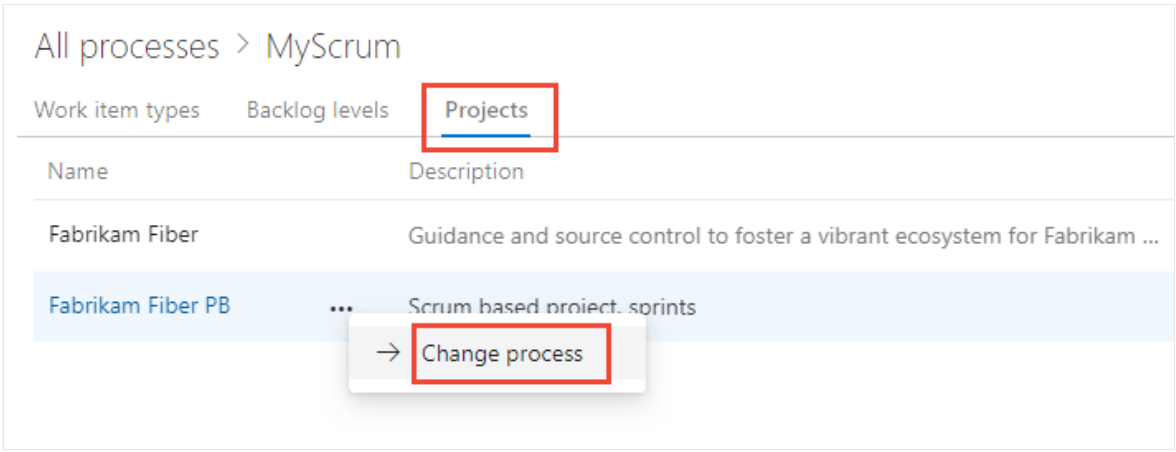


Change the process

1. Choose the process that contains the project you want to change. To change from Scrum to Agile, choose **Scrum** or a custom process based on Scrum.

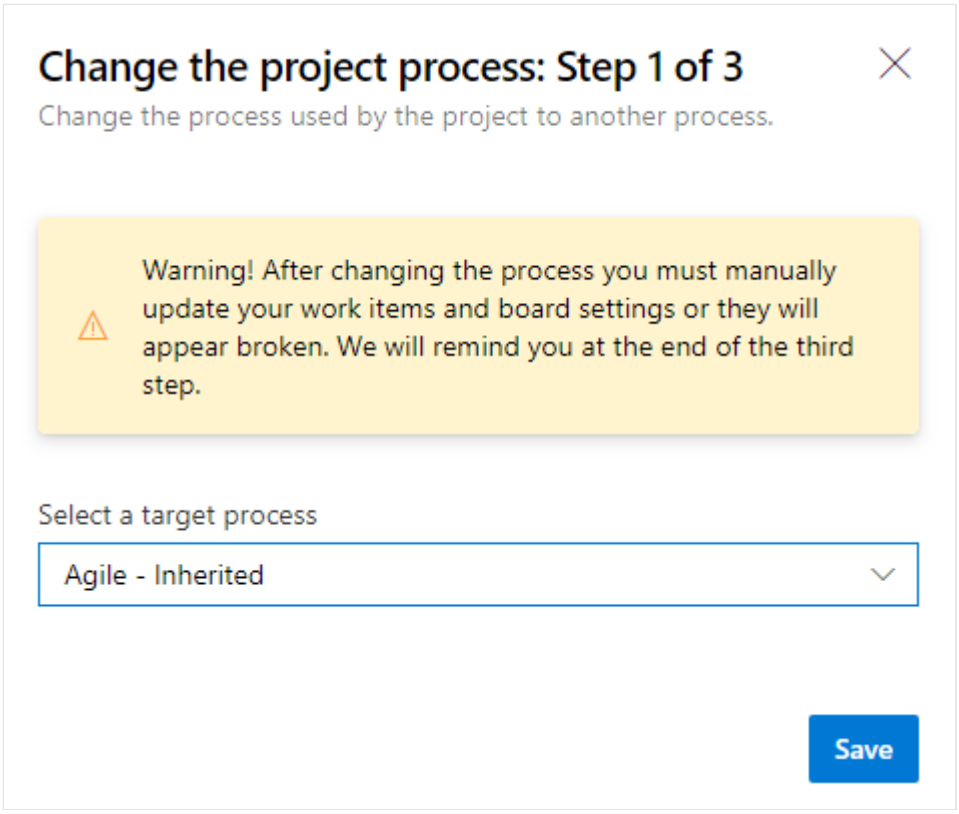


2. Choose **Projects**, and then choose the ··· actions icon for the project you want to change, and select **Change process**. Here we choose the Fabrikam Fiber PB project.

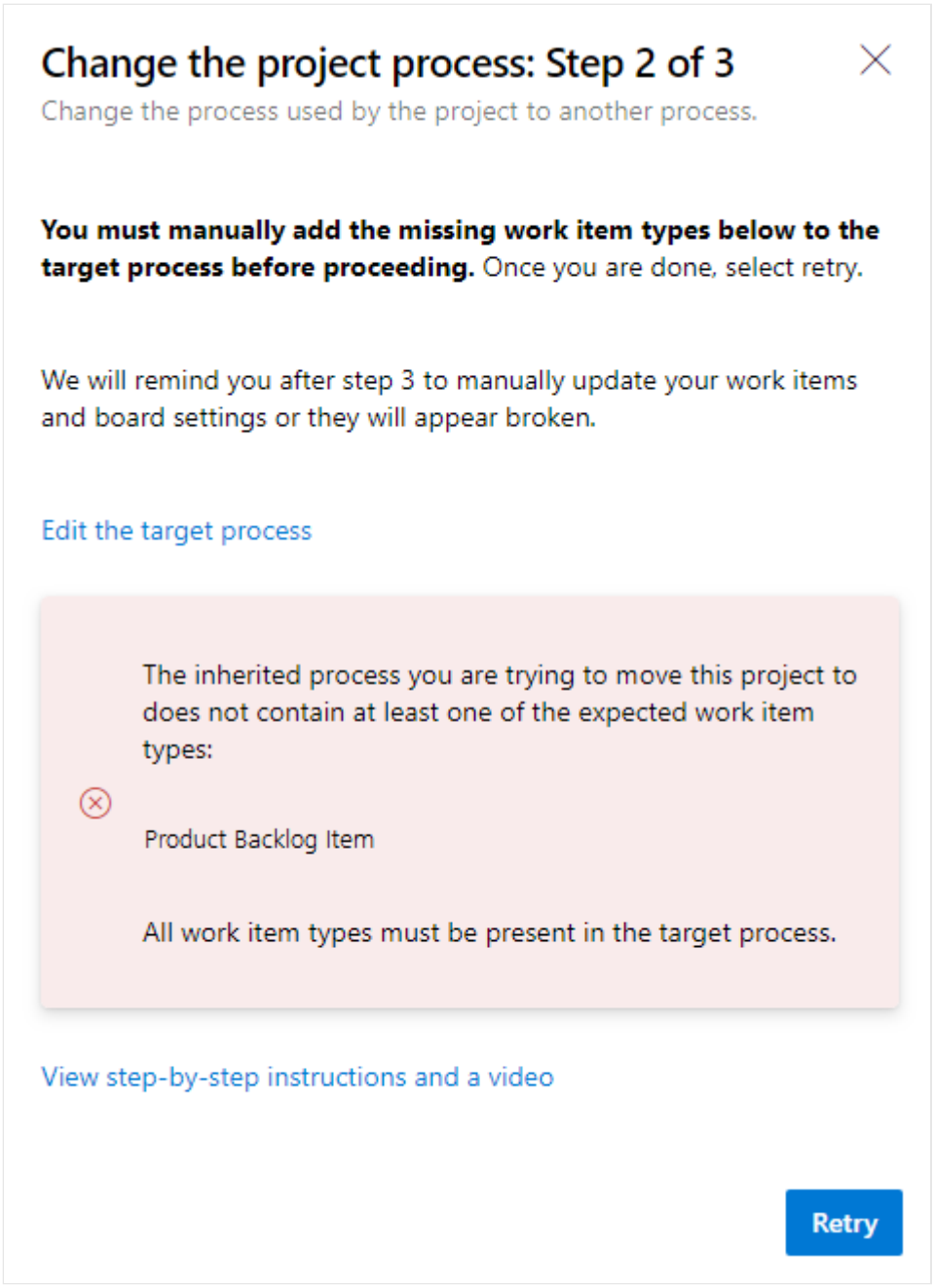


Follow the steps in the wizard.

3. Choose the **Agile - Inherited** process that you created earlier and then choose **Save**.

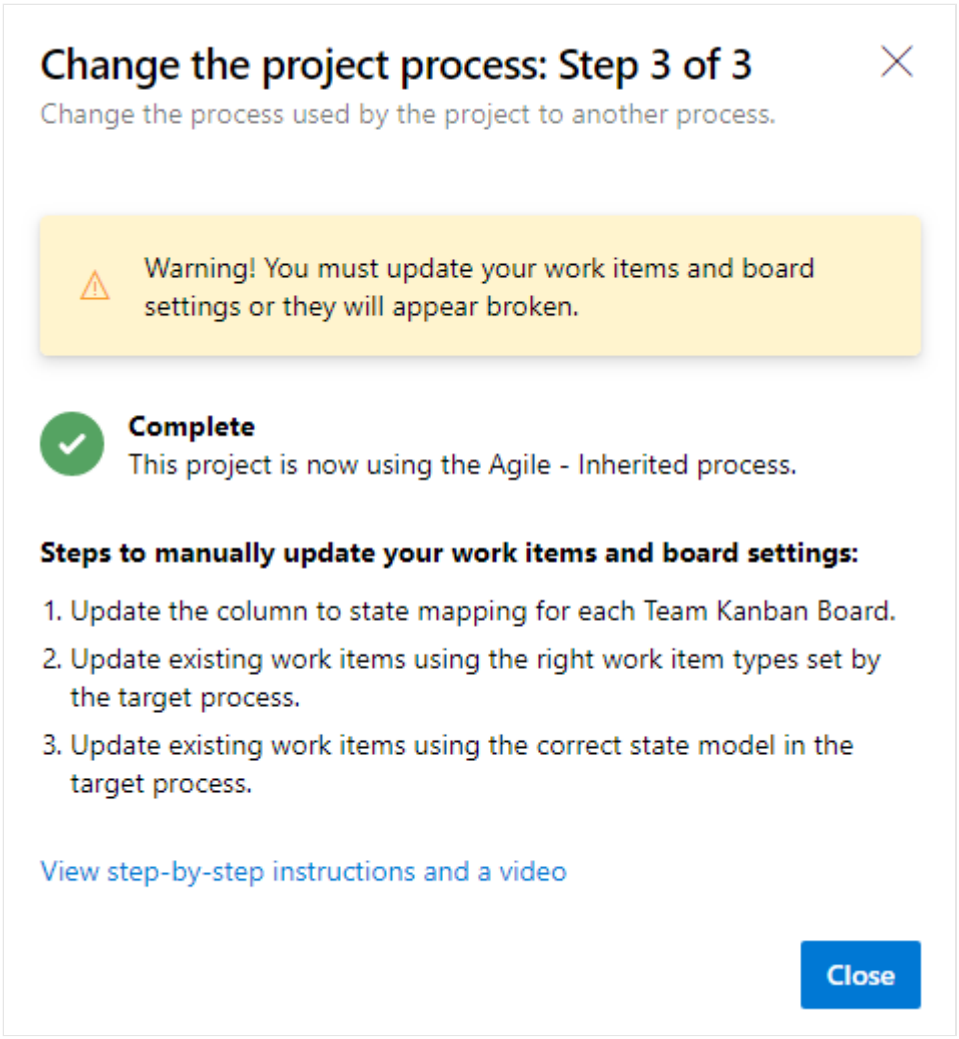


If you skipped step 2 of Create and customize an Agile inherited process, you'll receive the following message to add the missing work item types.



Close the wizard dialog and add the missing work item types now. See Add and manage work item types.

4. Upon successful completion, the wizard displays the following information. Make a note of the steps to follow and then choose **Close**.



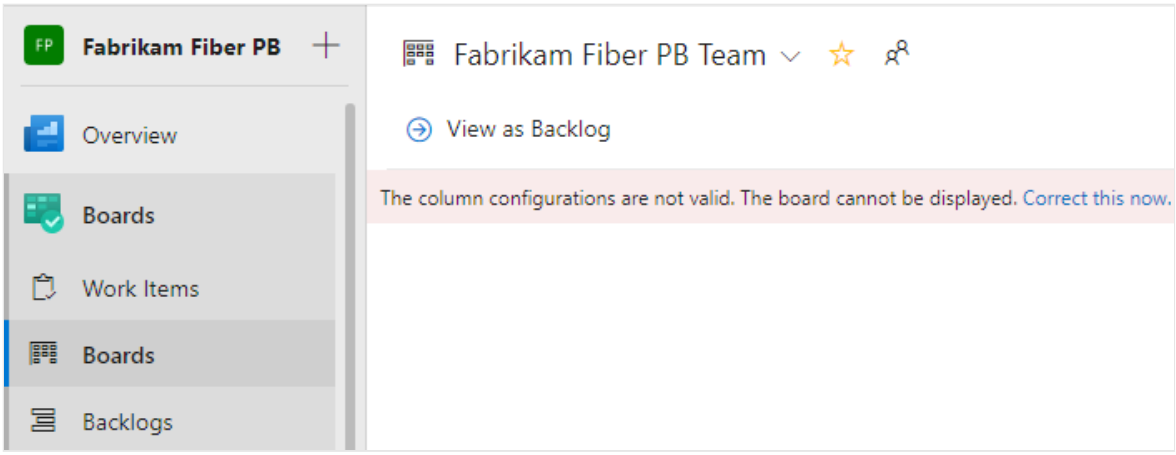
Steps to manually update your work items and board settings:

- Update the column to state mapping for each team Kanban board
- Update existing work items using the work item types set by the target process
- Update existing work items using the correct state model of the target process.

Update Kanban board column-to-state settings

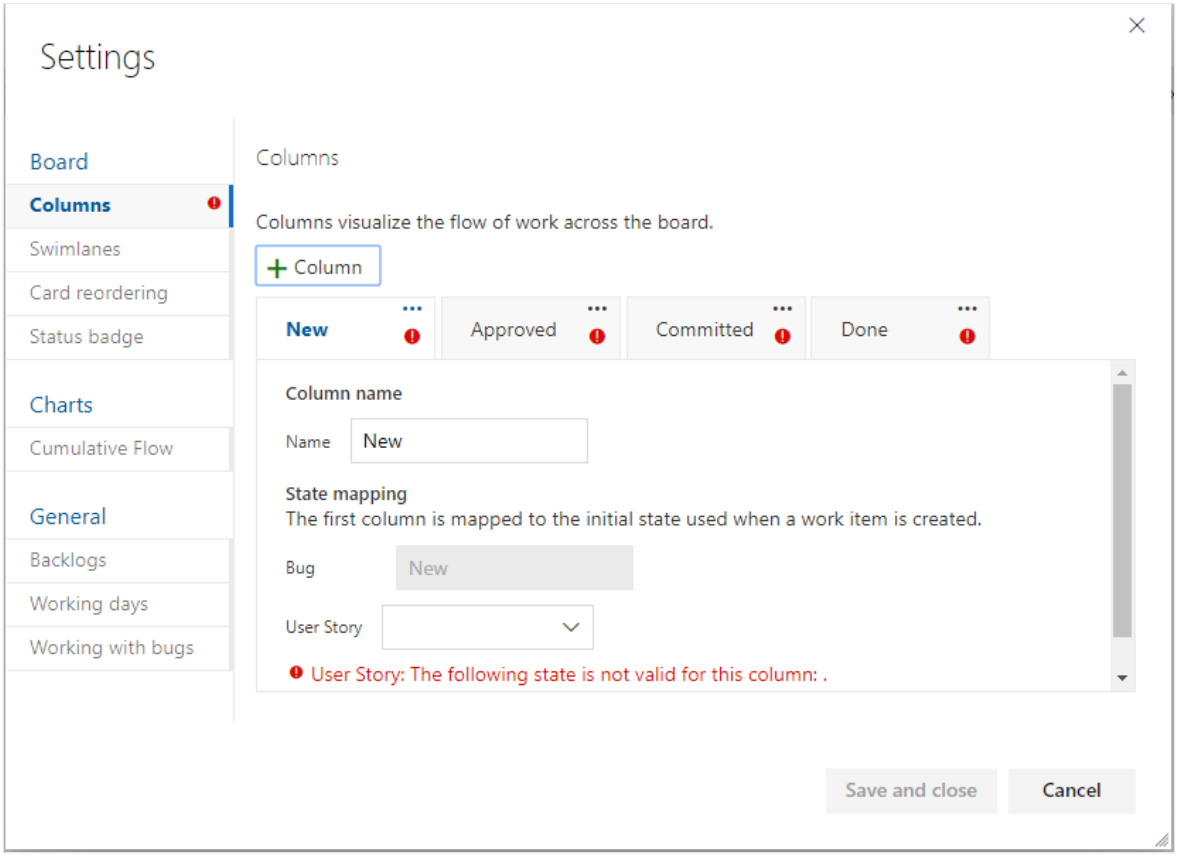
You can customize Kanban boards to display intermediate columns. For each column added, you must choose a valid workflow state for the work item types displayed on the board. To learn more, see Workflow states & state categories.

1. For each team, open your Kanban board.

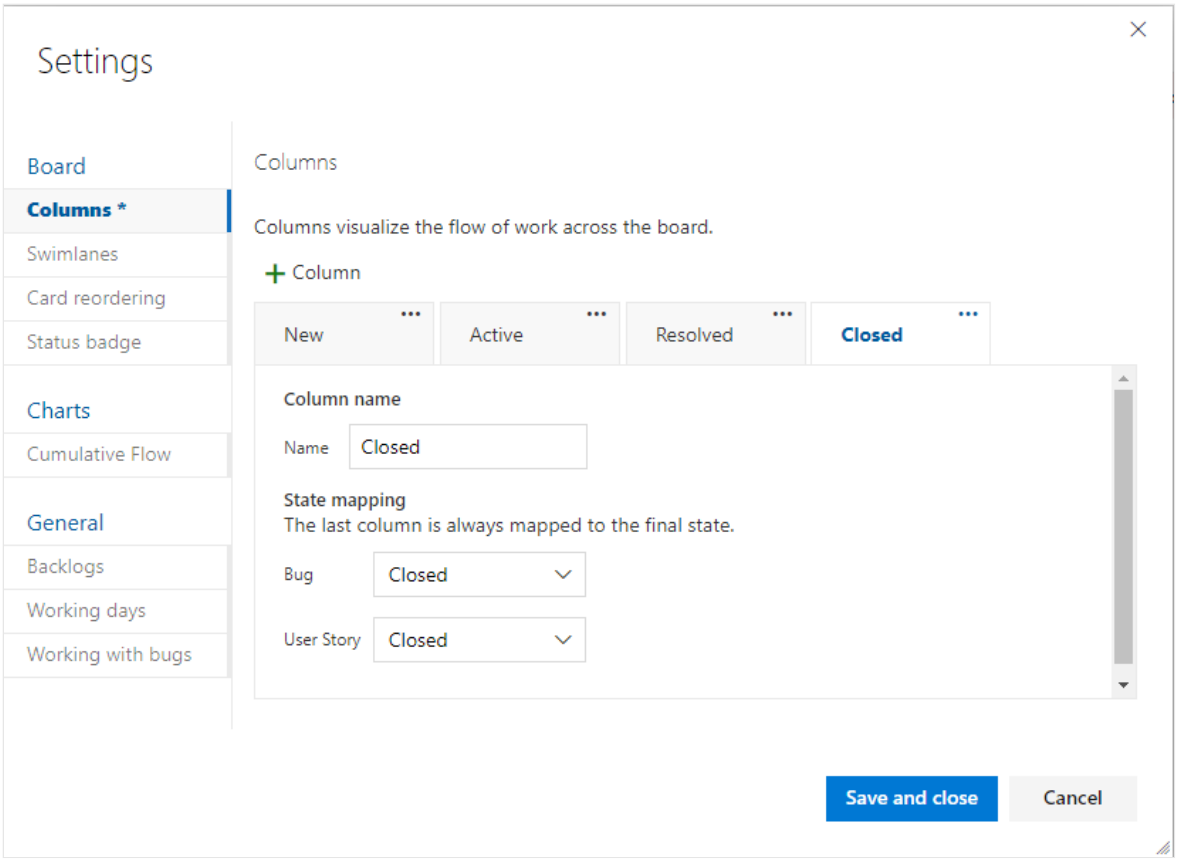


2. Choose the **Correct this now** link or the gear icon to configure the board settings.

   The Settings dialog opens. Those tabs that display a required icon need correction.



   Rename each column and choose the correct state for each column so that the column-to-state mapping is correct. As needed, add one or more columns. When done, choose **Save and close**.

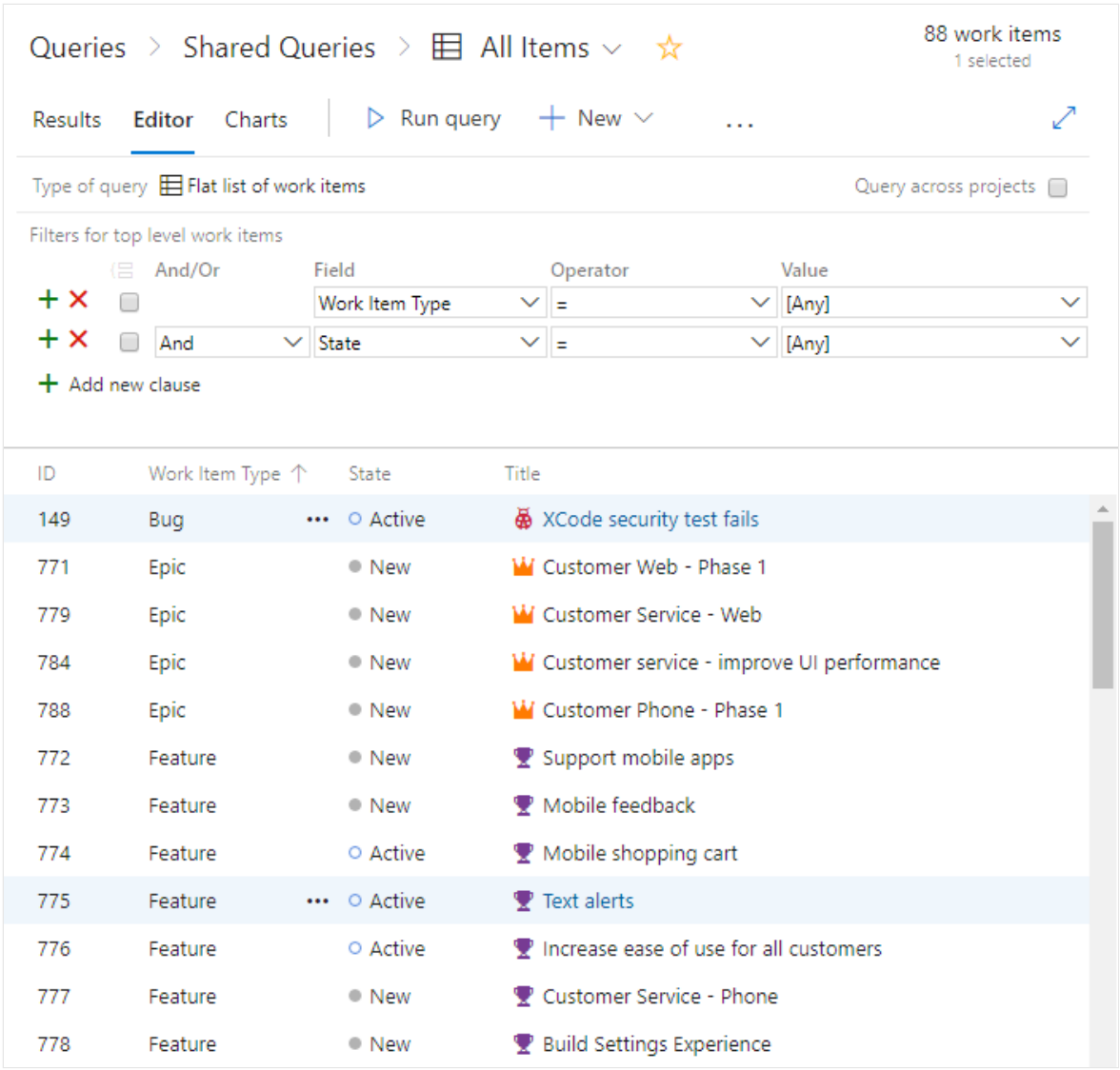


Update work items

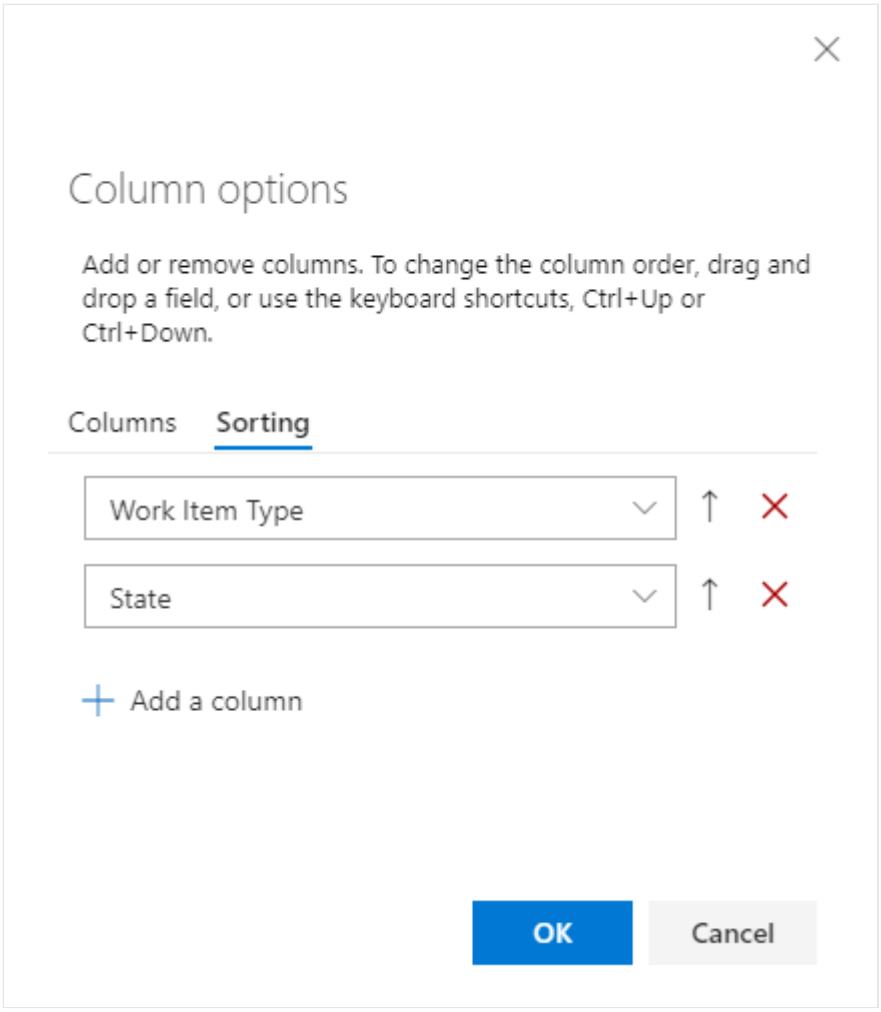
Your next step is to bulk update work items. The recommended sequence is:

- Create a work item query that displays all work items that need update.
- Perform a bulk update to change the work item type of Product Backlog Item work items to User Story.
- Perform a bulk update on all States to change from Scrum states—Approved, Committed, and Done—to Agile process states—Active, Resolved, and Closed.

1. Create a query to get a list of all Product Backlog Items, Impediments, Features, Tasks, and Epics. You will need to change the work item type of the first two work item types, and the State and Reason fields for all items.

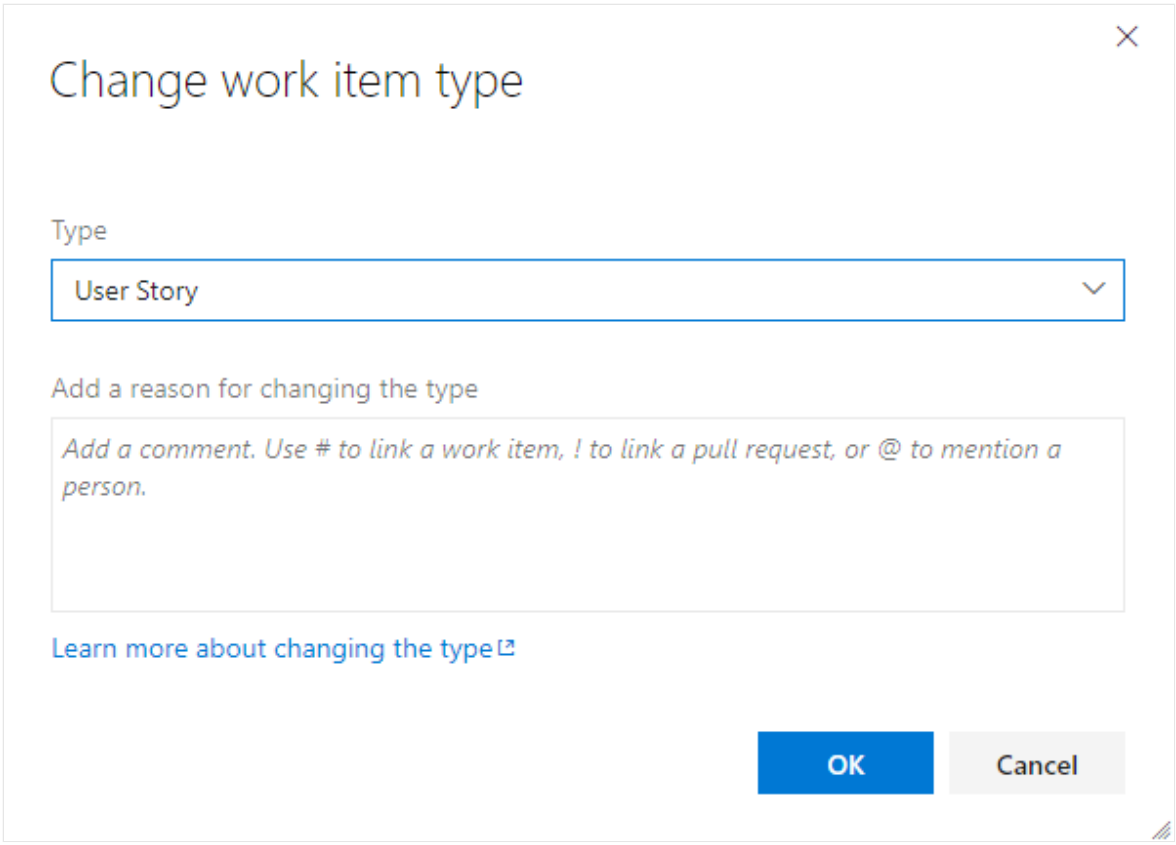


2. Choose the ··· actions icon and then select **Column options**. Choose to show the State and Reason fields. Choose the **Sort** tab, and set it to sort the list by work item type and state value.



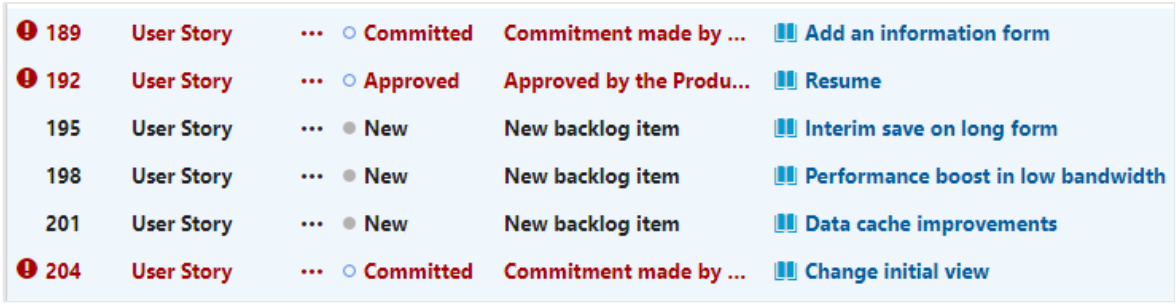
3. Choose **Results** to just show the list of work items.

4. Highlight all User Stories, choose the ··· actions icon, select **Change type**, and change the type to Product Backlog Item.

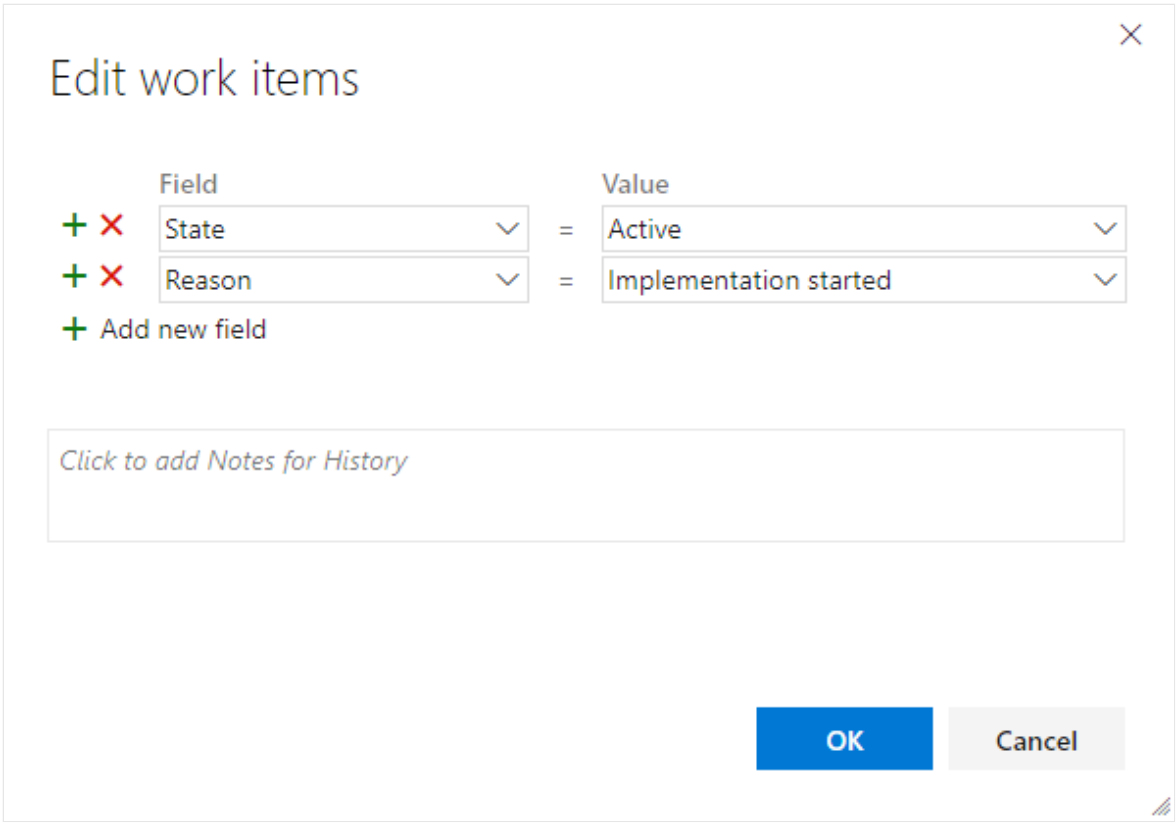


For more details, see Move, change, or delete work items, Change the work item type.

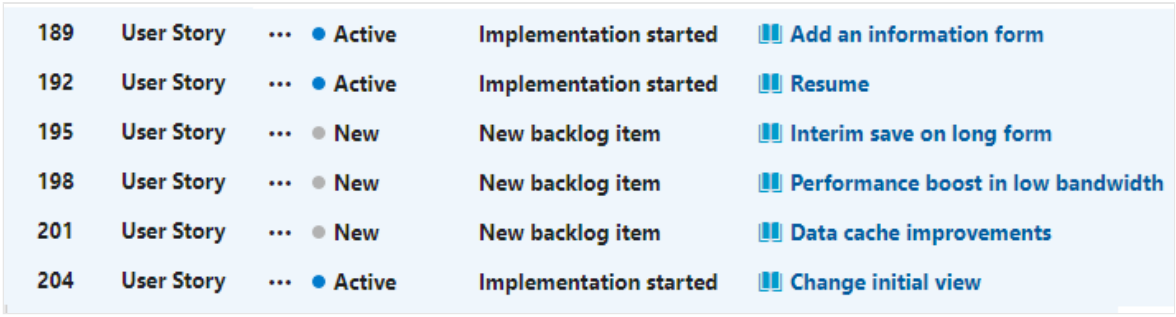
It's possible that you will receive errors where the work item type and the state are mismatched. In that case, you can't save your changes until you update the state as described in the next step.



5. Highlight all work items of the same State, such as Resolved, choose the ... actions icon, and then select **Edit**. Add the State field and select Committed or a valid value for the Product Backlog Item. You may also need to update the Reason field. For details, see Bulk edit work items.



This should result in an updated list without errors.

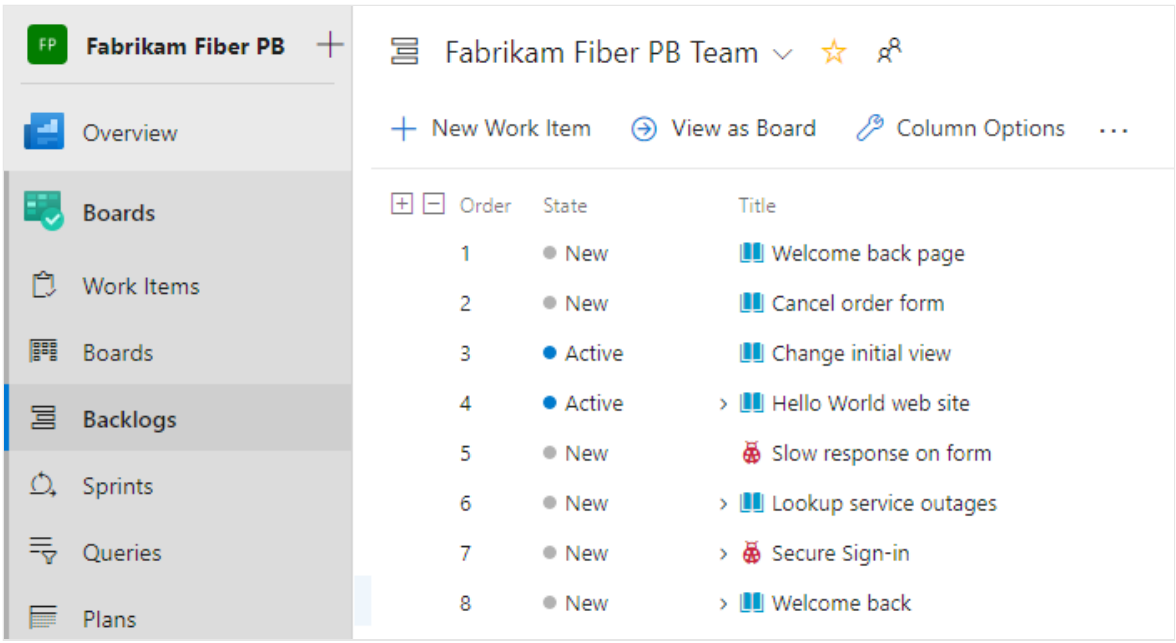


6. To save your changes, choose the ··· actions icon and select **Save items**.
7. Repeat the above steps to change the work item type for Impediment to Issue, and update the states to valid states for the work item types.
8. When done, make sure you choose the ··· actions icon and select **Save items**.

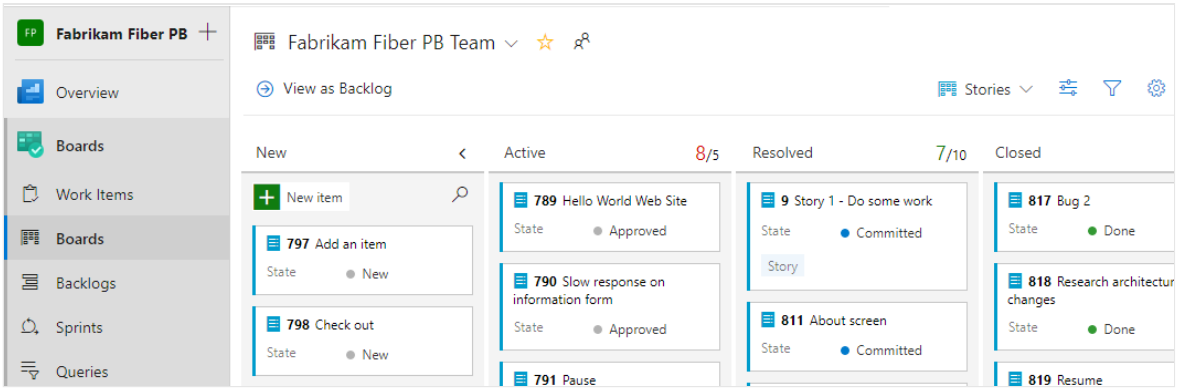
> **Tip**
>
> If you have a number of field value changes, such as populating the Story Points field with the value held in Effort, you can do that using Excel. See **Bulk add or modify work items with Excel**.

Verify your changes

1. Go to your team backlog and verify that the backlog items appear as expected.



2. Go to your team board and verify that the column settings are valid.



   To add columns or change column names, see Add columns to your Kanban board.

Optional updates

After changing the process, you may want to make additional updates as follows:

- Change the project name
- Update your project summary or vision
- Disable or delete the work item types that won't be used going forward. See Customize a work item type for an inherited process

Related articles

- Bulk modify work items
- Change the process used by a project

- Create an inherited process
- Add and manage work item types
- Show bugs on backlogs and boards
- Customize your boards
- Create and saved managed queries with the query editor

Change a project process from Basic to Agile

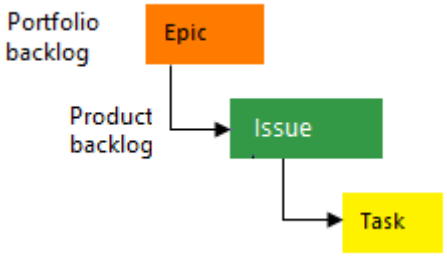
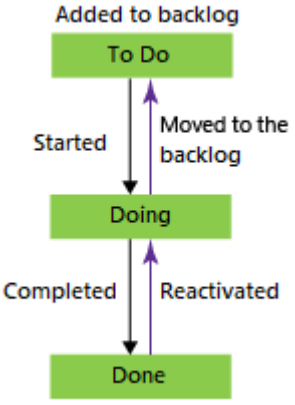
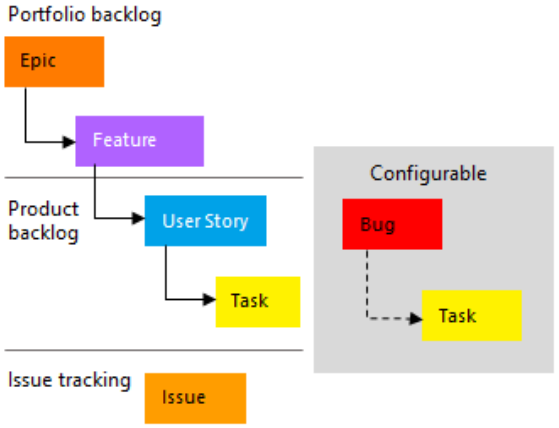
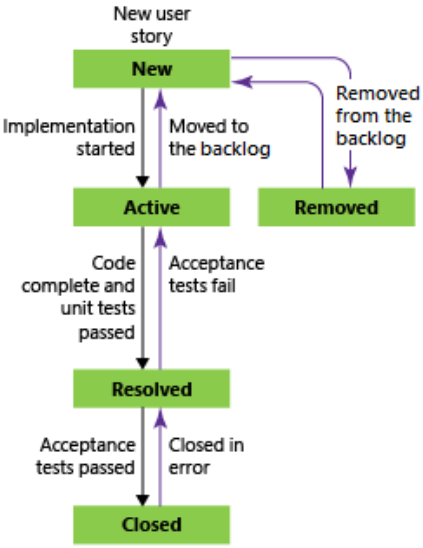
Article • 07/16/2024

Azure DevOps Services | Azure DevOps Server 2022 | Azure DevOps Server 2020

If you want to switch from the Basic process to an inherited Agile process for your project, follow the steps in this article.

Before you change your process, you should learn about the new process you're switching to. The Basic and Agile processes have the same Task and Epic work item types, but most of the values for the State and Reason fields are different.

Expand table

Process	Work item types	Workflow
Basic		
Agile		

Let's break down some of the benefits of transitioning your process from Basic to Agile:

- **Separate bug tracking:** In the Agile process, you can create separate work items specifically for tracking code defects (bugs). This action allows you to manage them independently from other work items like issues and user stories.
- **Agile workflow states:** Unlike the Basic process, Agile introduces different workflow states. These states align with Agile principles and help you manage work items more effectively. For example, you have states like "To Do," "In Progress," "Done," and others.
- **Portfolio backlogs:** Agile provides access to both Feature and Epic portfolio backlogs. These backlogs allow you to organize and prioritize work items at a higher level, making it easier to plan and manage your project.
- **Customized inherited process:** If your organization requires a customized process based on Agile principles, transitioning to Agile ensures compliance. You can tailor the process to your team's specific needs while still following Agile practices.

For more information, see About processes and process templates.

Prerequisites

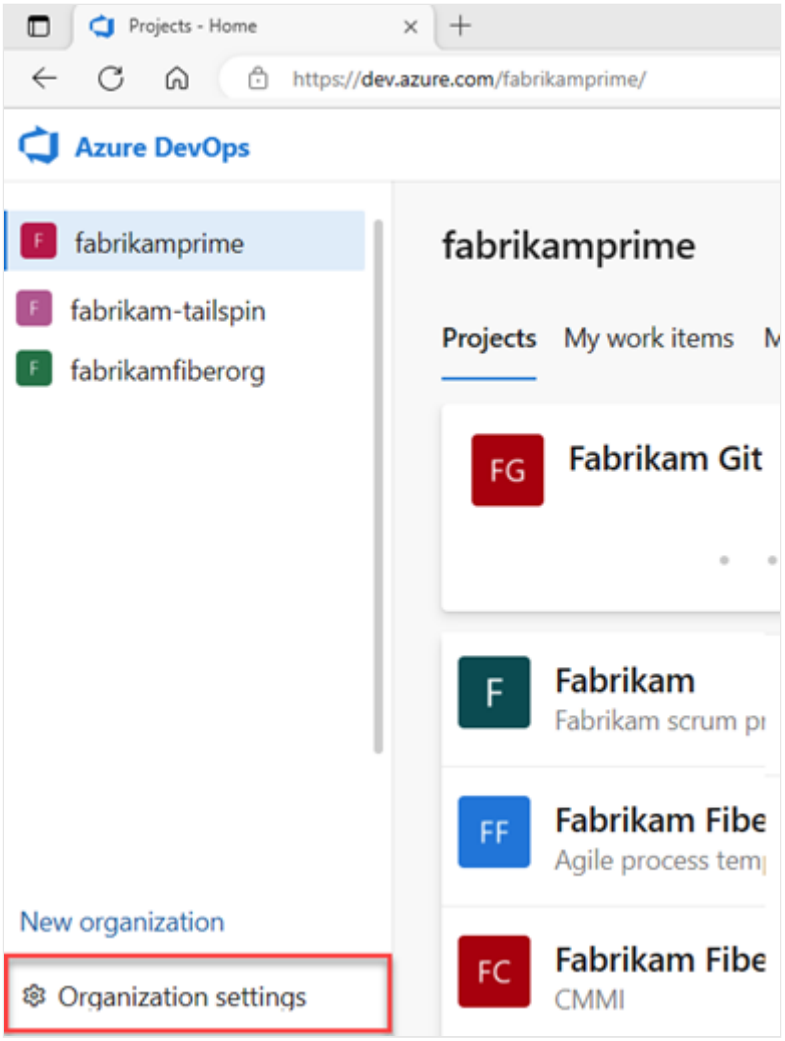
- To create, delete or edit a process, you must be a member of the **Project Collection Administrators** group, or have the collection-level permissions **Create process**, **Delete process**, **Edit process**, or **Delete a field from organization**. For more information, see Set permissions and access for work tracking, Customize an inherited process.
- Even if you have **Basic** or **Stakeholder** access, you can still change a process if someone gives you permissions to do so.
- To update boards, you must be the team administrator or a member of the **Project Administrators** group
- To update and change the type of your existing work items, you must be a member of the project.

> **Note**
>
> The images you see from your web portal may differ from the images you see in this article. These differences result from updates made to Azure DevOps. However, the basic functionality available to you remains the same unless explicitly mentioned.

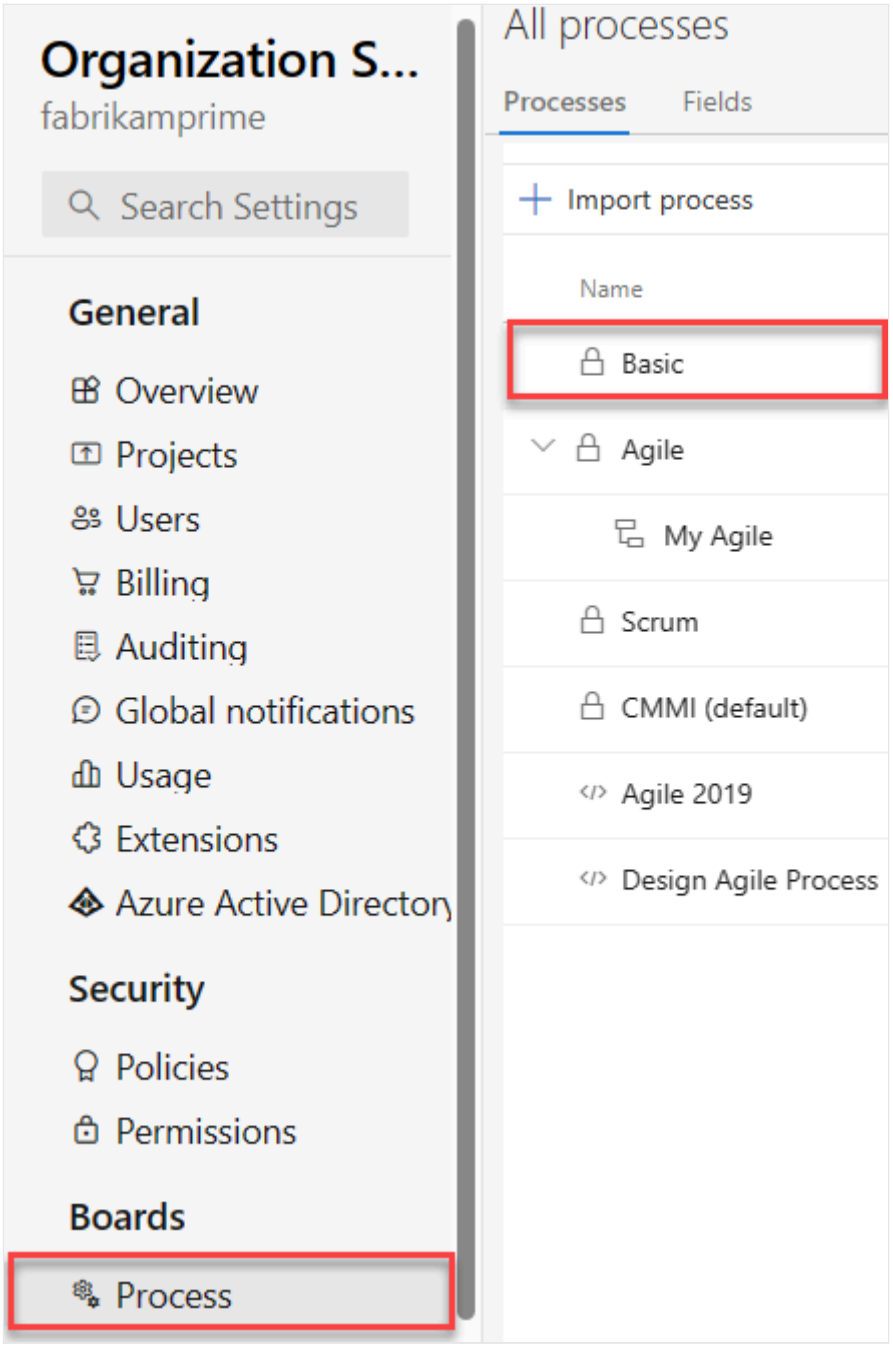
Change the process

1. Sign in to your organization: `https://dev.azure.com/{Your_Organization}`.

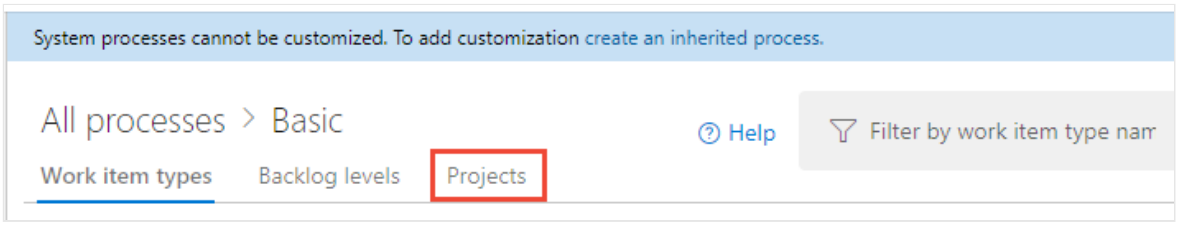
2. Select **Organization settings**.



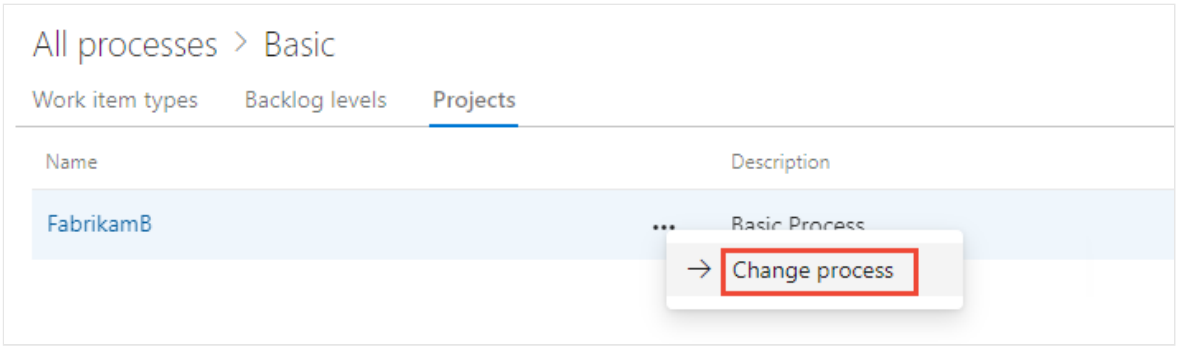
3. Select **Process**, and then choose **Basic**.



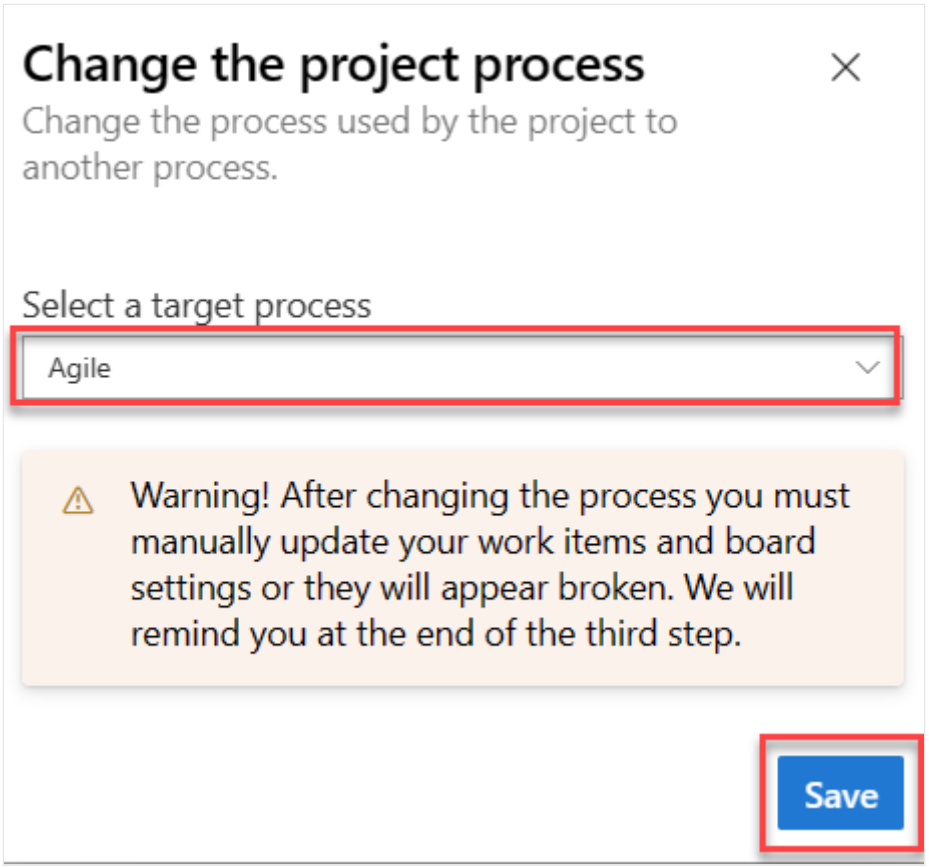
4. Select **Projects**.



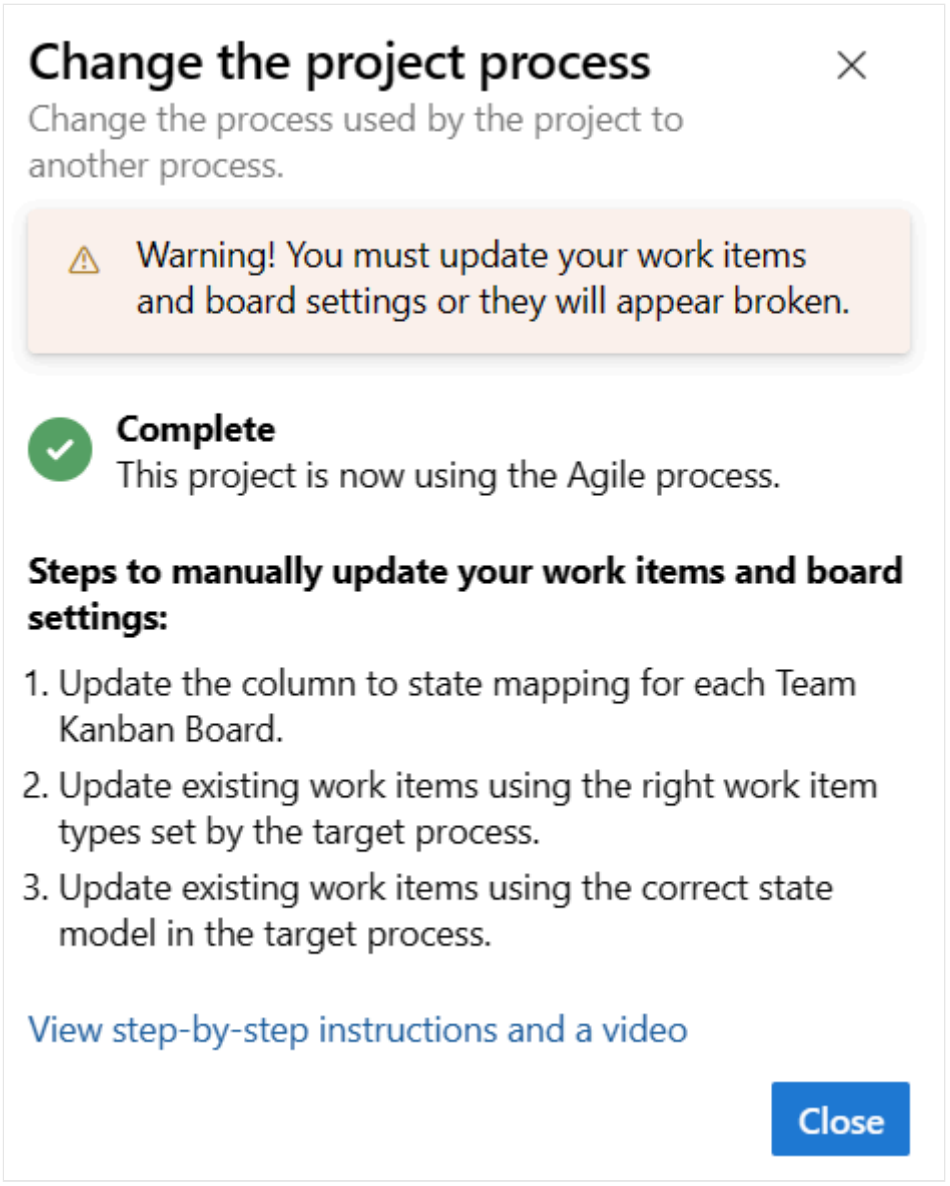
5. For the project you want to change, select the ··· actions icon and select **Change process** and follow the steps in the wizard.



6. Choose **Agile** from the dropdown menu, and then select **Save**.



The wizard displays the following information. Notice the steps to manually update your work items and board settings and **Close** the window.



7. (Optional) If you had existing work items, manually update work items and board settings.
   a. Update the column to state mapping for each team board.
   b. Update existing work items using the work item types set by the target process.
   c. Update existing work items using the correct state model of the target process.

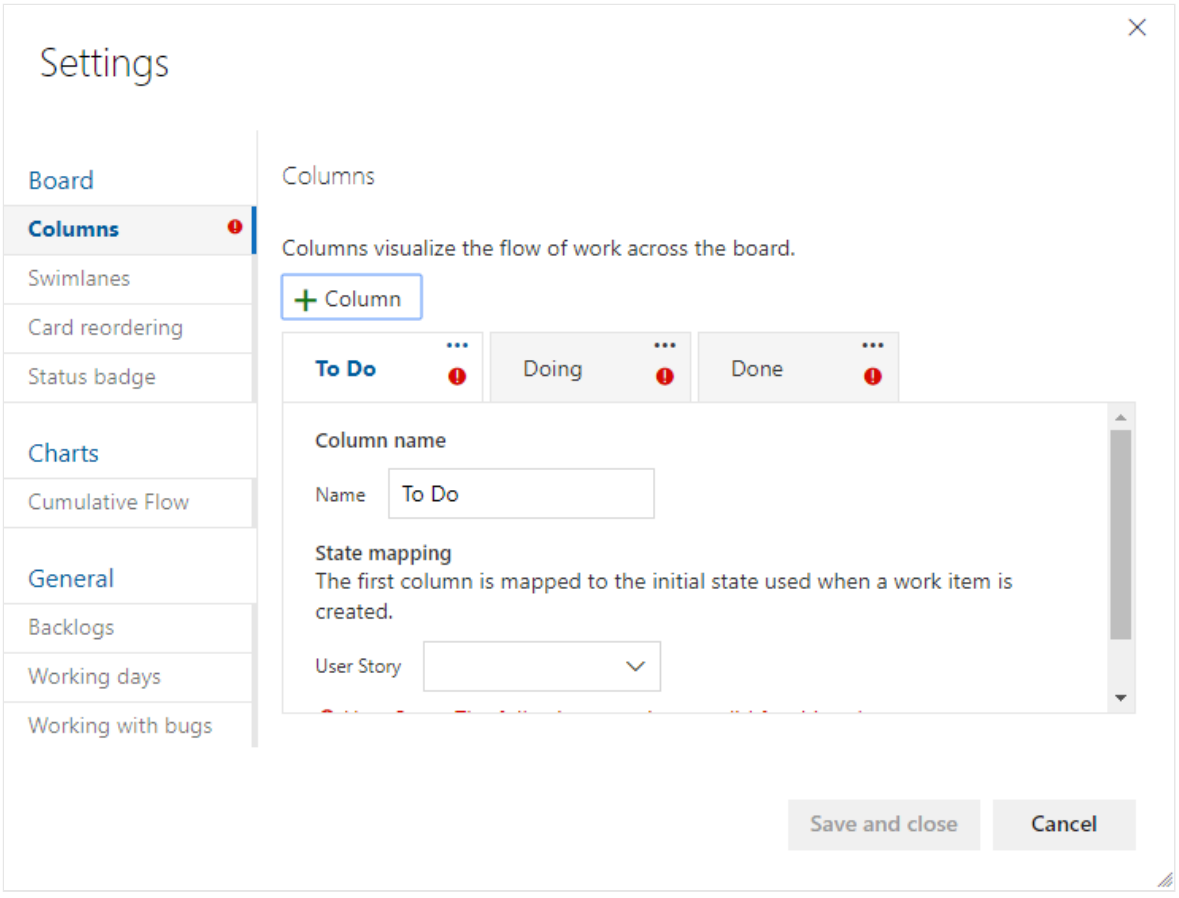
Update board column-to-state settings

You can customize boards to display intermediate columns. For each column added, you must choose a valid workflow state for the work item types displayed on the board. To learn more, see Workflow states & state categories.

1. For each team, open your board, and then select **Correct this now** or select the gear icon to configure the board settings.



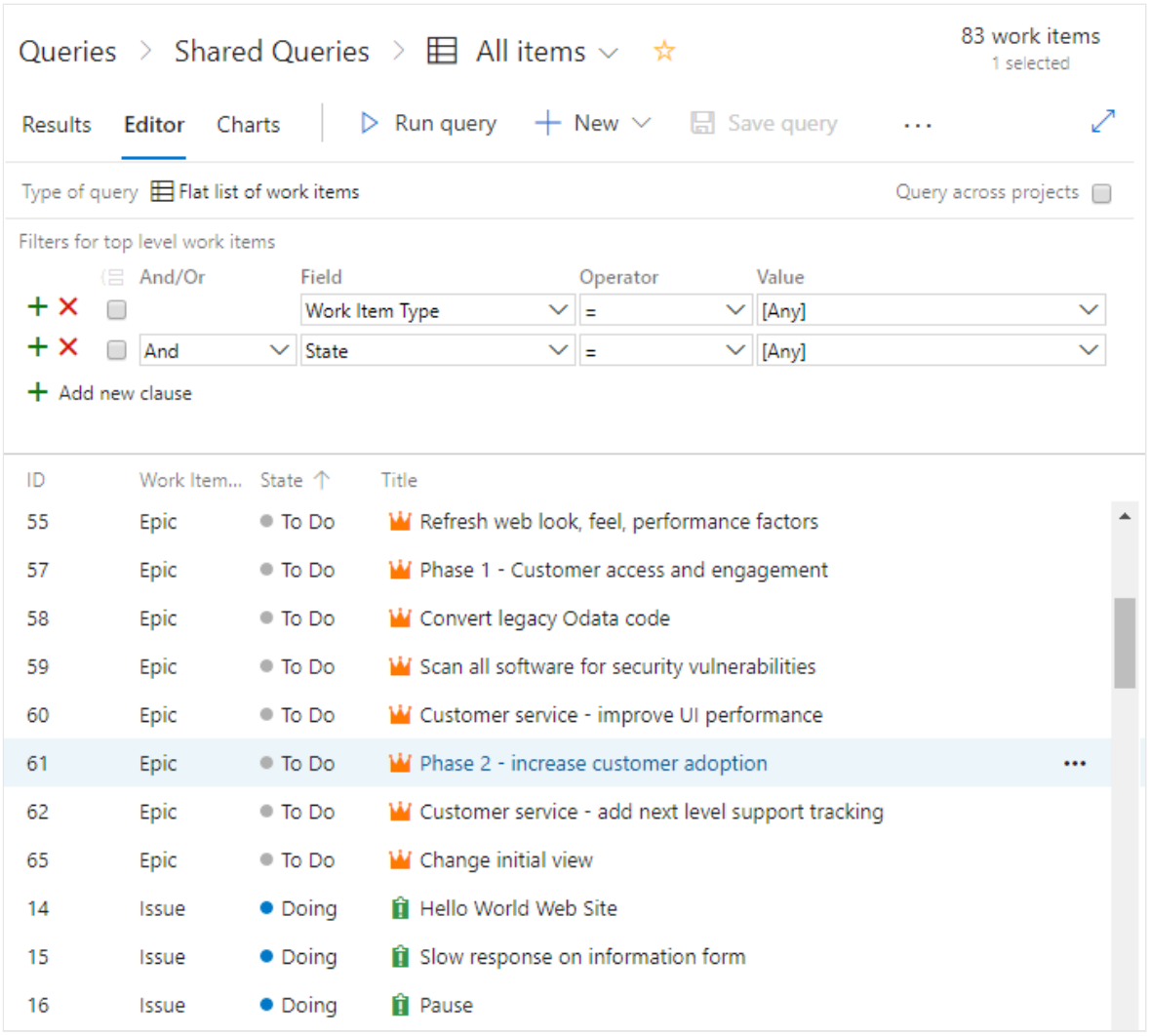
The Settings dialog opens. Those tabs that display a required icon need correction.



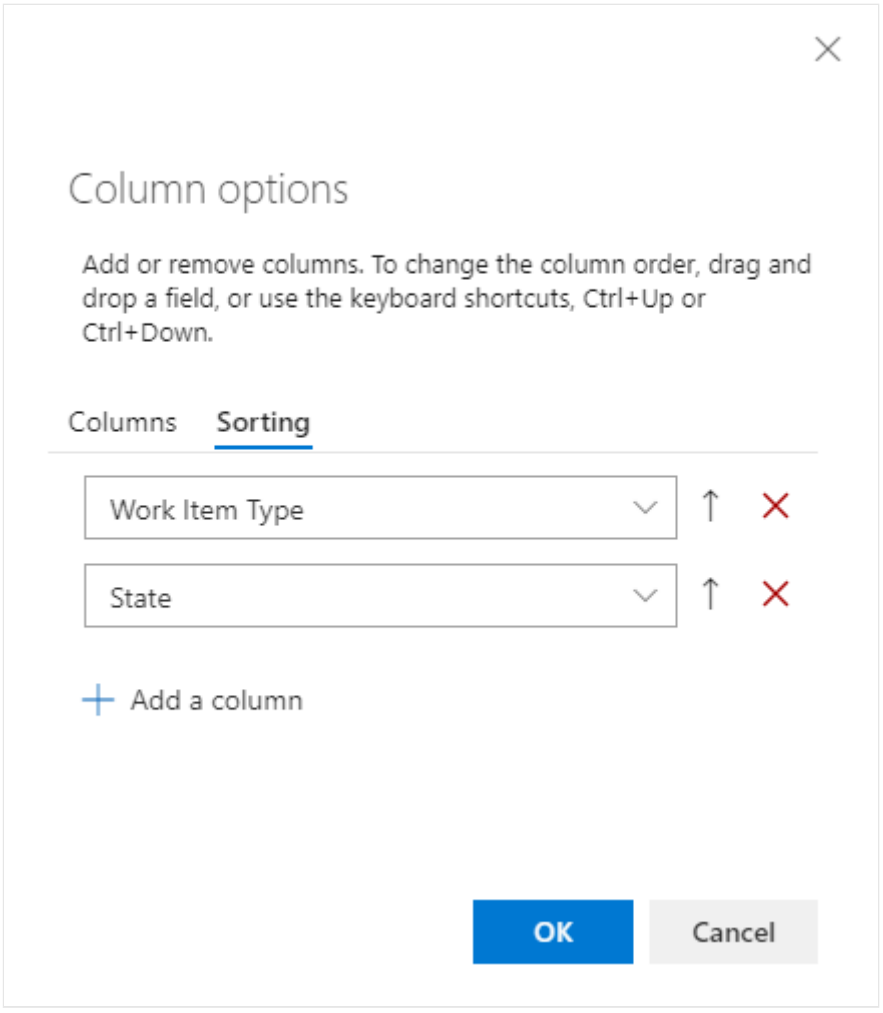
2. Rename each column and choose the correct state for each column so that the column-to-state mapping is correct. As needed, add one or more columns. **Save** your work. For more information, see Manage columns on your board.

Update work items

1. Create a query to get a list of all Issues, Tasks, and Epics.



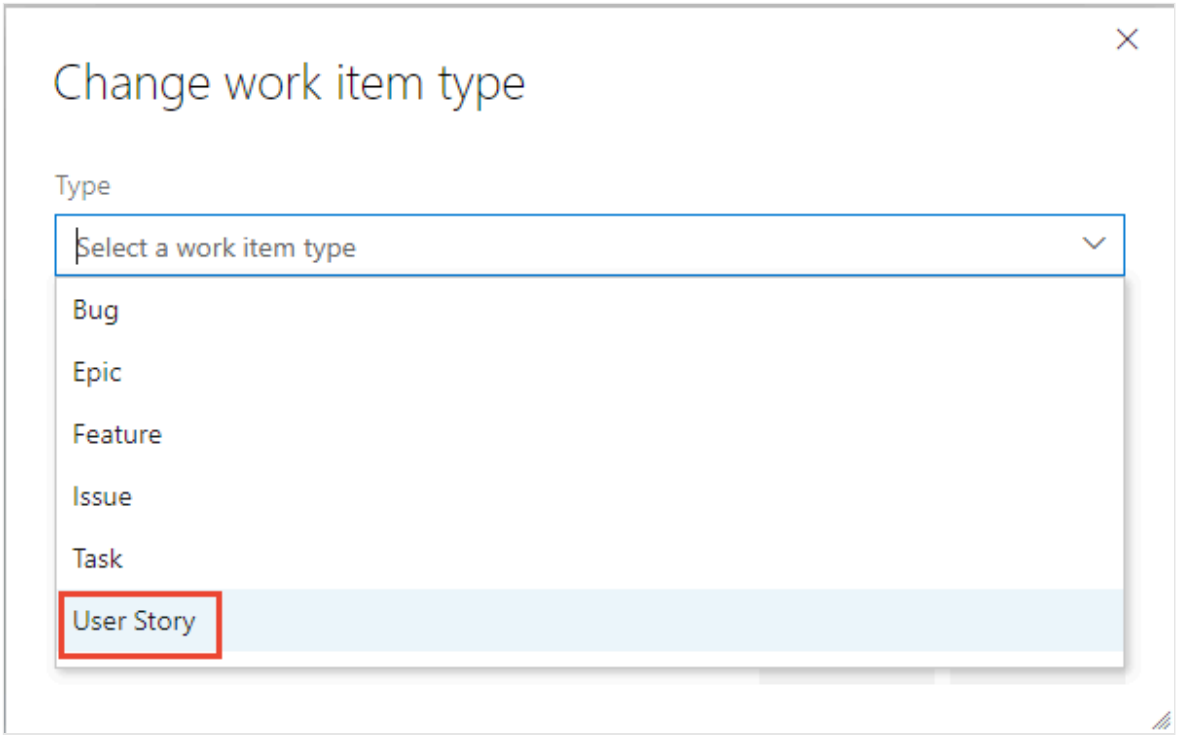
2. Edit your columns to be sure you show the State and Reason fields. Select the **Sorting** tab, and then set it to sort the list by work item type and state value.



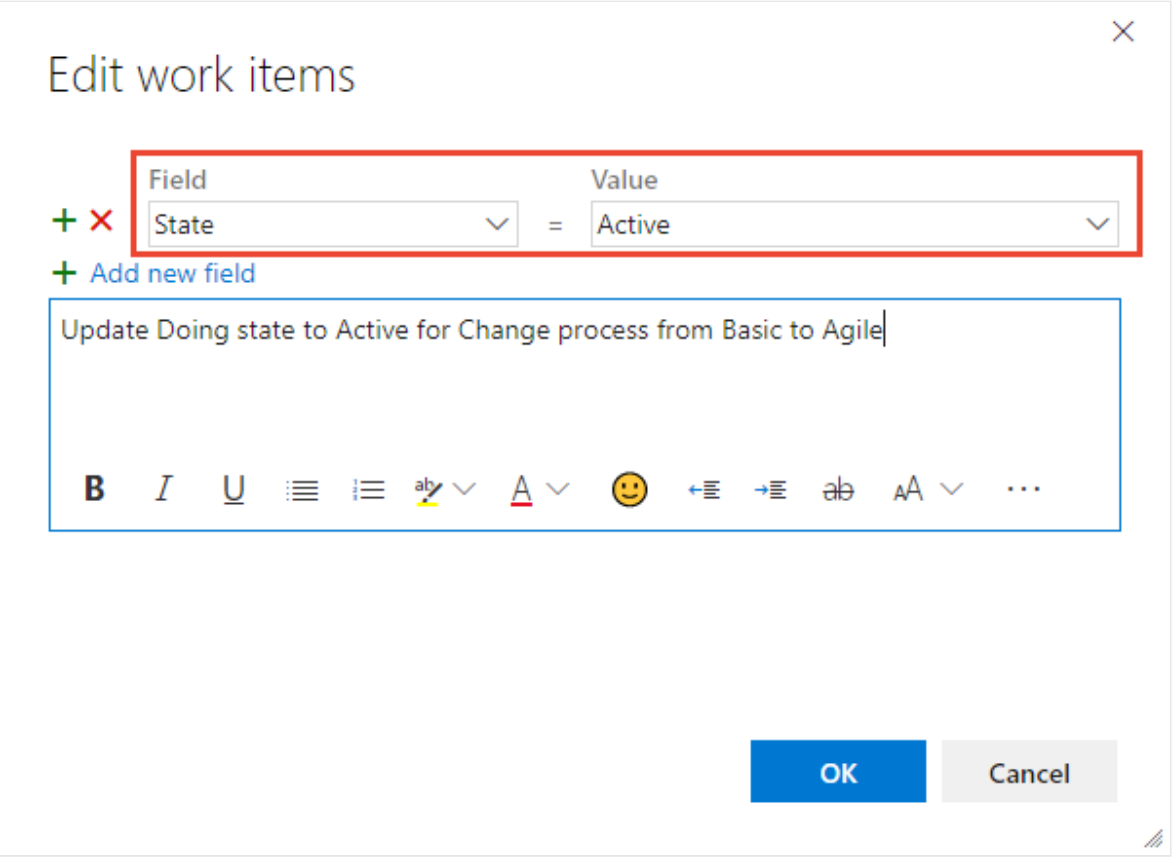
For more information, see Bulk add and modify work items in Excel.

3. Select **Results** to show the list of work items.

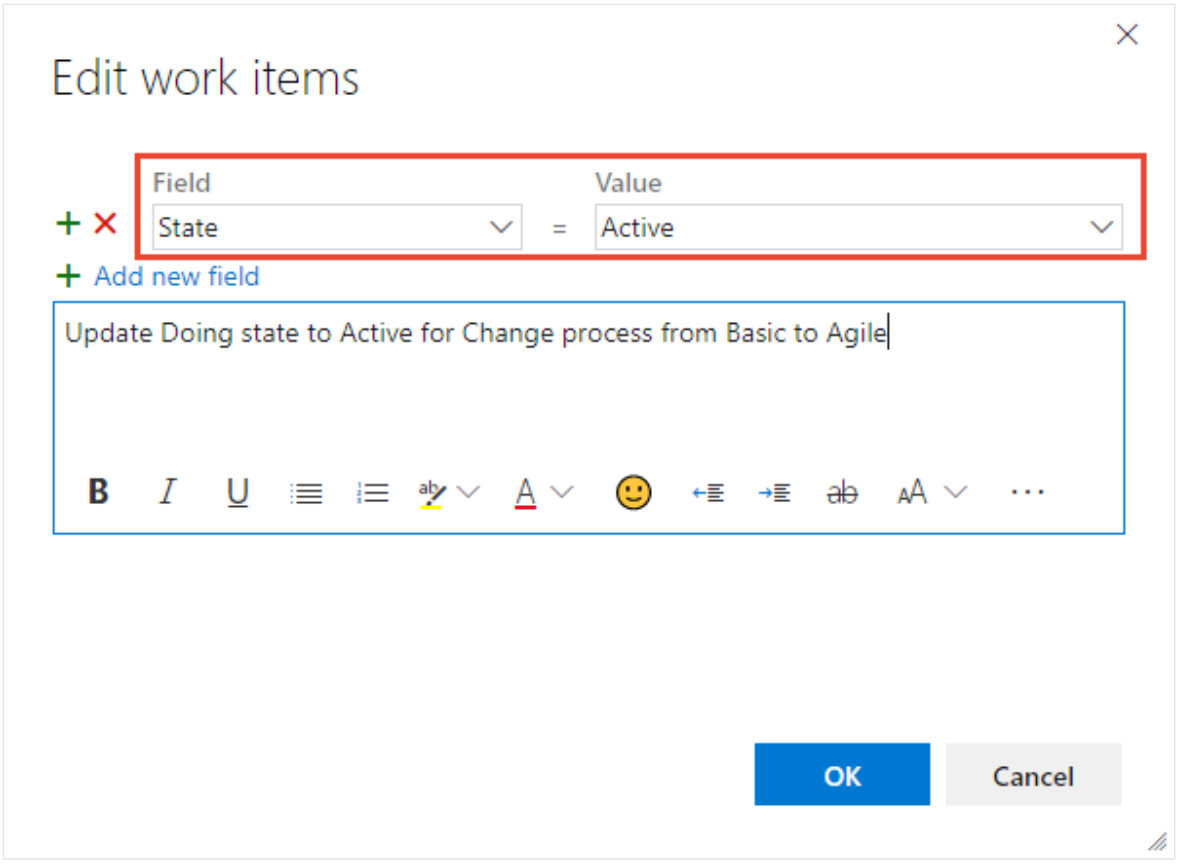
4. Highlight all Issues, choose the ··· actions icon, select **Change type**, and change the type to User Story and **Save**.



You might receive errors where the work item type and the state are mismatched. In that case, you can't save your changes until you update the state, which is described in the next step.



5. Sort the work items by the State column, highlight all work items of the same State, such as Doing, choose the ··· actions icon, and then select **Edit**. Add the State field and select Active for the value. For more information, see Bulk edit work items.

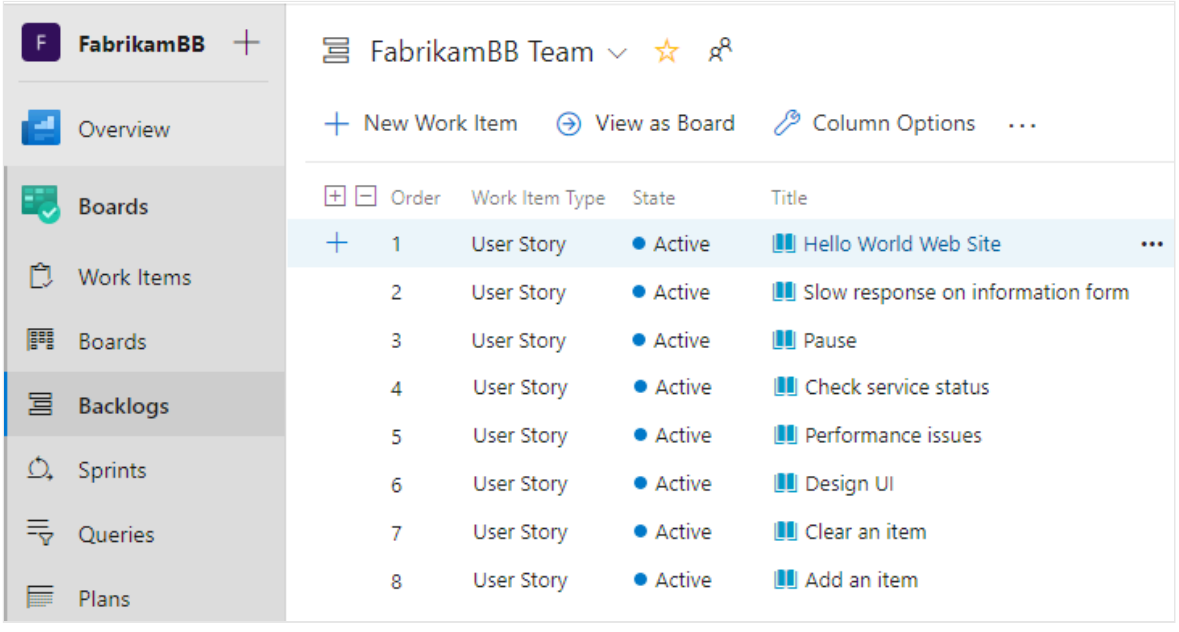


6. Repeat these steps for the Done state, changing it to Closed; and the To Do state, changing it to New.

7. **Save** your changes.

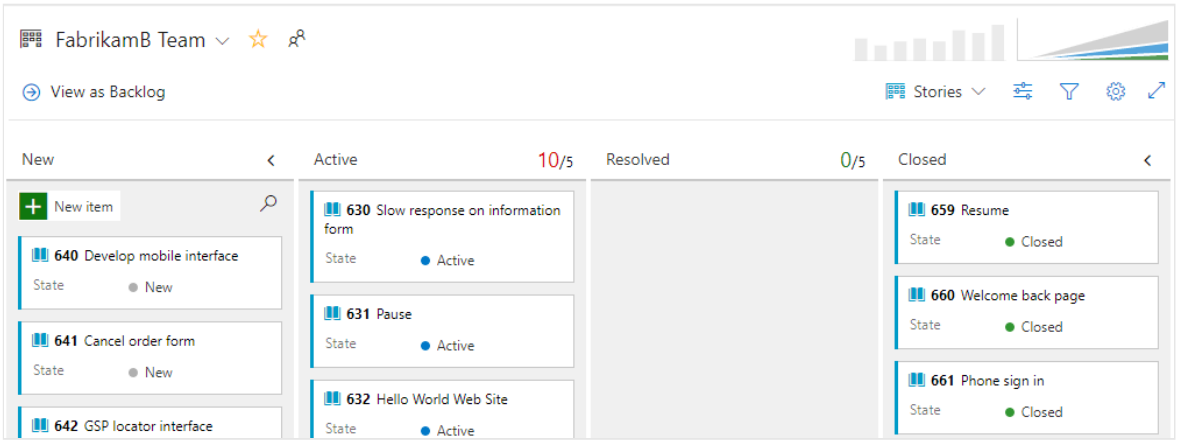
Verify your changes

1. Go to your team backlog and review the user stories.



To change any user stories to bugs, do so now using bulk update and update the **Change type**. If you want to show bugs at the same level as user stories, make that change now. For more information, see Show bugs on backlogs and boards.

2. Go to your team board and verify that the column settings are valid.



Related articles

- Modify work items in bulk
- Create an inherited process
- Add and manage work item types
- Show bugs on backlogs and boards
- Create and saved managed queries with the query editor

Feedback

Was this page helpful? Yes No

Provide product feedback

List work item fields and attributes

Article • 11/22/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Work item fields are used to track information within Azure DevOps. These fields are defined at the organization level and are shared across all projects in the organization. To review the fields defined for your organization, you can use one of the following two tools, available for both Inherited and Hosted XML process models:

- Process > Fields web page
- Work Item Field Explorer

For a description of each field defined with a system process, see Work item field index.

Prerequisites

Permissions: To view the fields defined for an organization or collection, be a member of the **Project Collection Valid Users** application group or have the **View instance-level information** permission set to **Allow** for the organization or collection.

List or review fields

To list or review fields, use one of the following tools based on your process model—**Inherited**, **Hosted XML**, or **On-premises XML**. For an index of fields defined within the default processes, see Work item field index.

Expand table

Tool	Inheritance	Hosted XML	On-premises XML
Web portal: List inherited and custom-defined fields	✓	✓1	
Work item field explorer	✓	✓	✓
witadmin listfields command line tool	✓	✓	✓

> **Note**

1. Only supported for default processes (Agile, CMMI, Scrum).

Field data types and names

Each work item type specifies the fields included in work items of that type. Each field has multiple attributes, many of which are system-defined and immutable.

Fields get defined by the following three attributes:

- **Data type**: Indicates the kind of data that can be entered into the field, such as Boolean, Double, Integer, HTML, or String. For detailed descriptions of each data type, see Query fields, operators, and macros.
- **Friendly name**: The user-friendly name assigned to the field, used when selecting a **Field** in a query clause. This name might differ from the name displayed on the work item form.
- **Reference name**: The identifier used when creating WIQL queries, improvised work item templates, executing REST API commands, or defining XML work item type definitions. The reference name is permanent and can't be changed once set.

For a detailed description of each field attribute and instructions on how to list them, see Field attributes and List field attributes later in this article. For an overview of work item types (WITs) and work items, refer to Track work with user stories, issues, bugs, features, and epics.

What is a field? How are field names used?

Each work item type includes 31 system fields and several type-specific fields. Work items are used to plan and track your project.

Each field captures specific information about the work to be performed. The values you assign to these fields are stored in the work tracking data store, enabling you to create queries that determine status and trends.

For descriptions and usage of each field defined for the core system processes—Agile, Basic, Scrum, and CMMI—refer to the Work item field index.

Field names

A work item field name uniquely identifies each work item field. Ensure your field names adhere to the following guidelines:

- **Uniqueness**: Field names must be unique within the account or project collection.
- **Length**: Field names must contain 128 Unicode characters or fewer.
- **Spacing**: Field names can't have leading or trailing spaces or contain two or more consecutive spaces.
- **Composition**: Field names must include at least one alphabetic character.
- **Prohibited Characters**: Field names can't include the following characters: `.,;'`:~/*|?"&%$!+=()[]{}<>`.

Since custom fields are defined at the organization or collection level, you can't add a custom field with the same name to multiple processes.

For more information, see Naming restrictions and conventions.

System and predefined fields

A work item field name uniquely identifies each field within Azure DevOps. Ensure your field names comply with the following guidelines:

- **Uniqueness**: Field names must be unique within an account or project collection.
- **Length**: Field names must be 128 Unicode characters or fewer.
- **Spacing**: Field names can't have leading or trailing spaces or contain multiple consecutive spaces.
- **Composition**: Field names must include at least one alphabetic character.
- **Prohibited Characters**: Field names can't include the following characters: `.,;'`:~/*|?"&%$!+=()[]{}<>`.

Since custom fields are defined at the organization or collection level, you can't add a custom field with the same name to multiple processes.

For more information, see Naming restrictions and conventions.

Custom fields

Because custom fields are defined at the organization or project collection level, you can't add a custom field with the same name to multiple processes.

When adding custom fields, note the following limits:

- **Maximum fields per work item type (WIT):** 64
- **Maximum fields per process:** 512

The field data type determines the kind and size of data that can be stored in the field. Each field can have only one type defined within a project collection, encouraging the

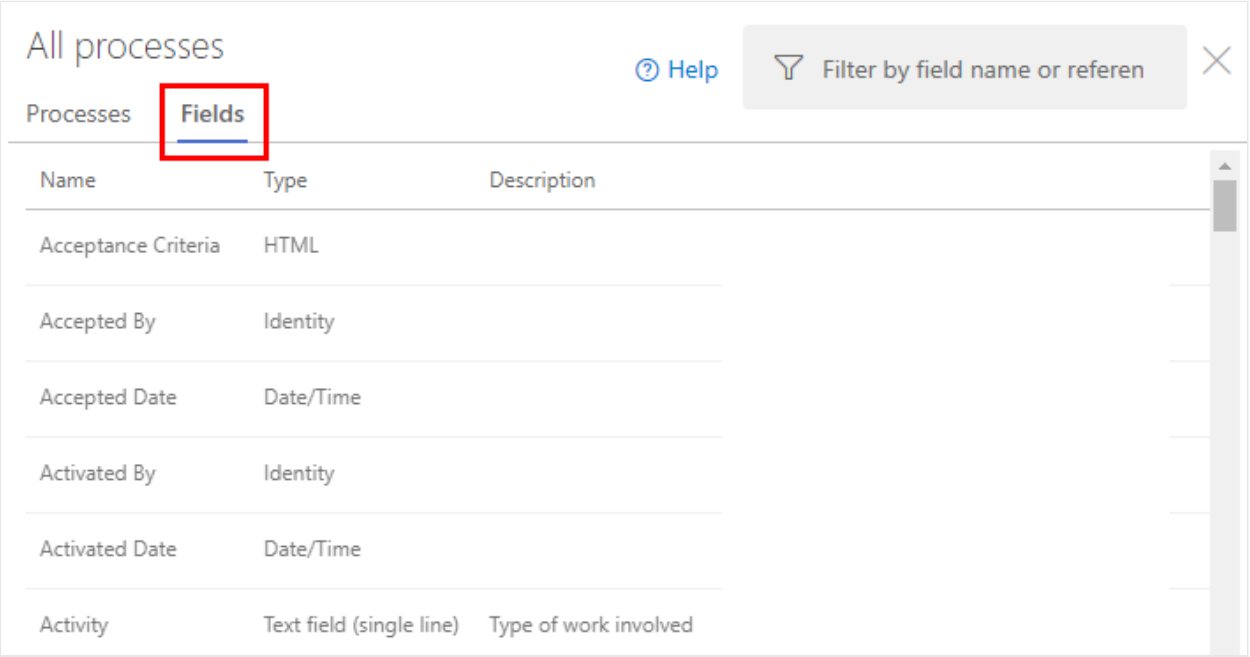
use of common fields across different projects and work item types.

When you add a custom field to an inherited process, Azure DevOps assigns a reference name prefixed with *Custom* followed by the field name without spaces. For example, adding a field named *DevOps Triage* results in the reference name **Custom.DevOpsTriage**. Spaces aren't allowed in reference names.

For more information, see Naming restrictions and conventions.

How can I determine the field data type?

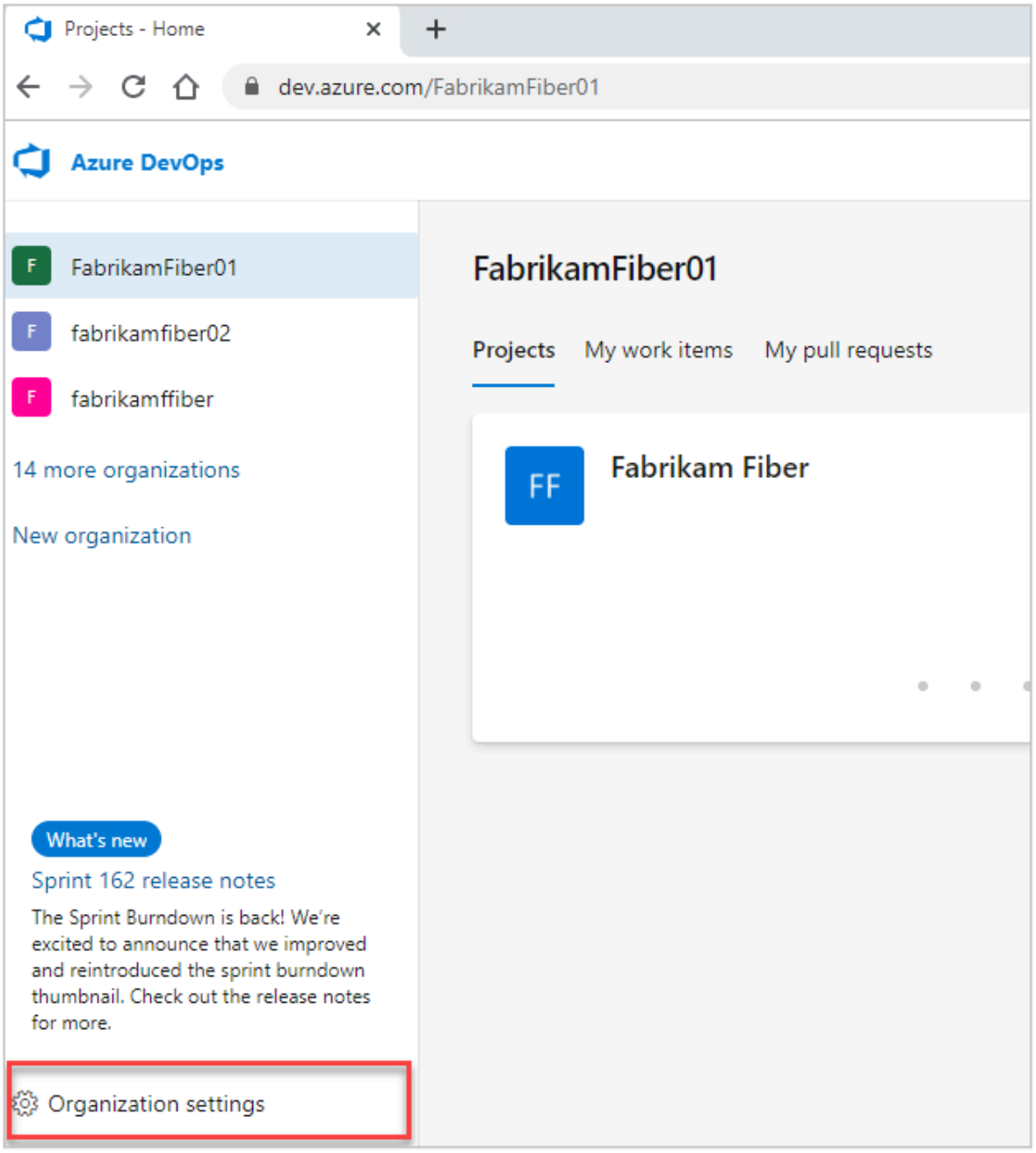
You can view the data type of fields defined for your organization by opening the Process>Fields page.



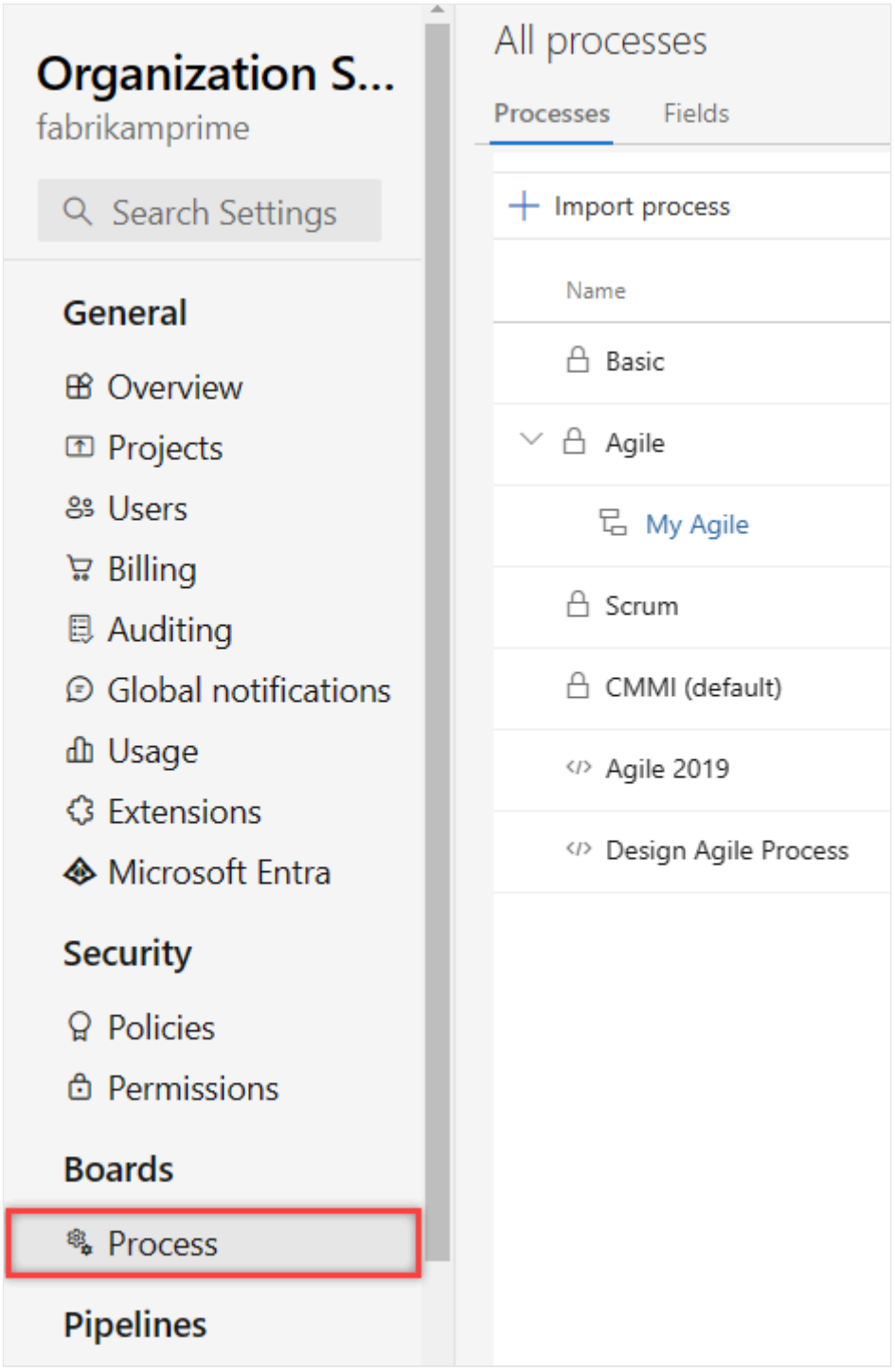
Review fields list

To review the list of fields defined for an organization or collection, do the following steps:

1. Select the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.

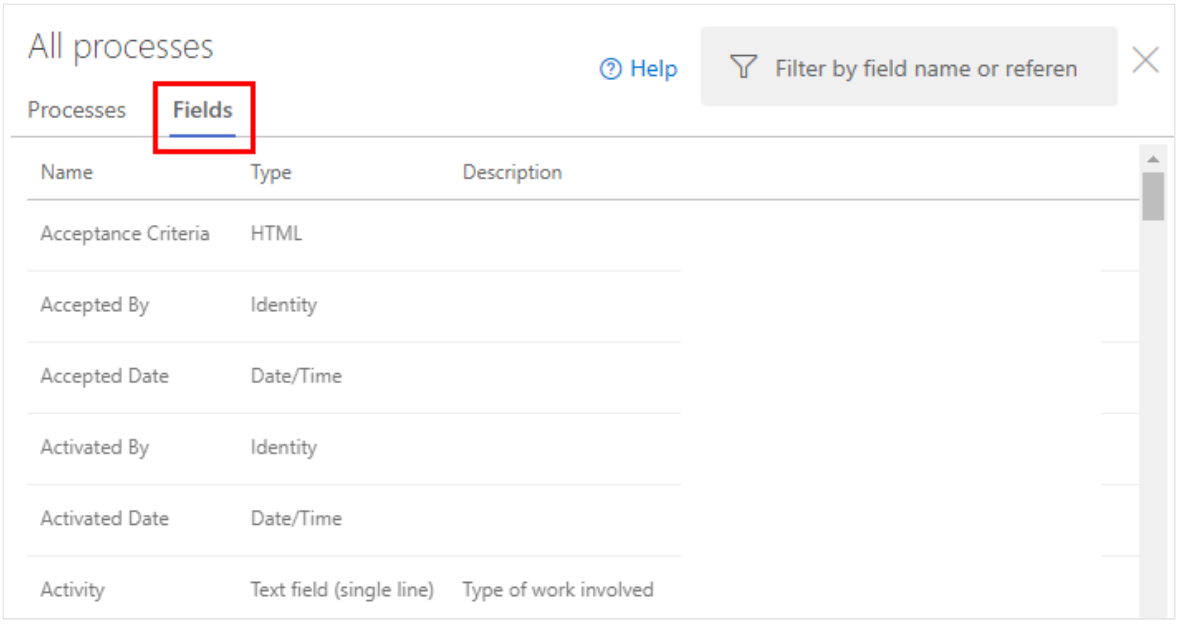


2. Select **Process**.



3. Select **Fields**.

   Fields listed correspond to all fields defined for the organization or collection, which includes all custom fields and the fields defined for system processes.

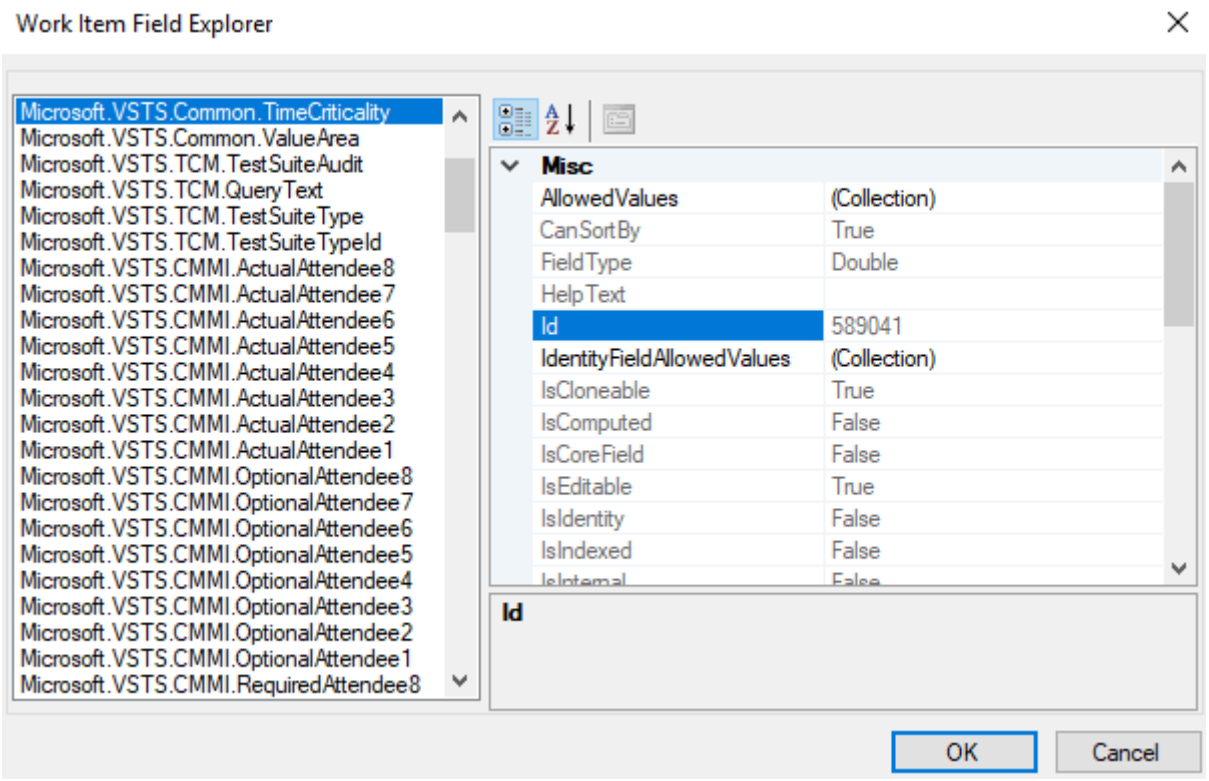


> **Note**
>
> If you don't see **Fields**, then your collection uses the On-premises XML process. The Fields page isn't supported for that process.

For descriptions, usage, and reference names of each field, refer to the Work item field index. Additionally, you can retrieve field reference names using the Work Item Types Field - List REST API.

Work Item Field Explorer

You can look up the assignments of field attributes using the Work Item Field Explorer tool.



To access the Work Item Field Explorer, install the Process Editor Tool. Based on your installed version of Visual Studio, get the Process Editor Tool from one of the following extensions.

- **Visual Studio 2019 & 2022:** As of now, there's no specific Process Template Editor extension available. But, you can still customize process templates using other tools such as any text editor or XML editor. You can also use the Azure DevOps web portal for certain customizations, especially for inherited processes.
- **Visual Studio 2017**: TFS Process Template Editor. You can also use this version of the Process Editor to modify the old-style work item forms. You can't use it to edit forms associated with the new web forms.
- **Visual Studio 2015**: TFS Power Tools.

Field attributes

There are many nonchangeable and hidden attributes for each work item field. The following table describes each attribute. Attributes have different names depending on whether you retrieve them through the **Fields - Get REST API** or view them through the Work Item Field Explorer (**WIFE**) tool.

Attributes assigned to a field depend on the platform and version you're using. For example, some attributes aren't supported with the Inheritance process. To look up the reference name for a field, see the Work item field index.

Attribute

Attribute type

Description

REST:
WIFE: **AllowedValues**

collection

Gets the collection of valid values for a field that contains picklist values. You can change this by specifying a picklist or global list (on-premises).
Can change?=Yes

REST: **canSortBy**
WIFE: **CanSortBy**

boolean

Indicates whether you can sort query results with this field.
Can change?=No

REST: **description**
WIFE: **HelpText**

string

Specifies a description for the field, which also defines the help text that appears when you hover over the field within the work item form.
Can change?=Yes

REST:
WIFE: **ID**

Integer

Specifies the internal ID of the field.
Can change?=No

REST:
WIFE: **IsCloneable**

boolean

Indicates whether the value defined for the field is copied when a user chooses to copy a work item. For example, **Title**, **Tags**, and **Description** fields are copied, but the **ID** and **History** fields aren't copied.
Can change?=No

REST:
WIFE: **IsComputed**

boolean

Indicates if the value set by this field is computed by the system (True) or not (False). Examples of computed fields are ones that are set by the system, such as the **ID**, **Revised Date**, **Changed Date**, and **External Link Count**.
Can change?=No

REST:
WIFE: **IsCoreField**

boolean

Indicates whether this field is specified for all work item types.
Can change?=No

REST:
WIFE: **IsEditable**

boolean

Indicates if users can modify this field (True) or not (False). Examples of noneditable fields are ones that are set by the system, such as the **ID**, **Revision**, **Created By**, and **Changed By** fields
Can change?=No

REST: **isIdentity**
WIFE: **IsIdentity**

boolean

Indicates whether this field is an *Identity* field. Identity fields are string fields used to store user identities.
Can change?=No

REST:
WIFE: **IsIndexed**[1]

boolean

Indicates whether this field is indexed to support search.
Can change?=No

REST:
WIFE: **IsLongText**

boolean

Indicates that the field can contain more than 255 characters, such as fields assigned a data type of *PlainText*, *HTML*, or *History*.
Can change?=No

REST: **isPicklist**[2] WIFE:

Boolean

Indicates whether the field is associated with a picklist. The value is set to *True* when a custom field is defined for Azure DevOps and Picklist (String) or Picklist (Integer) type is selected. The value is set to *False* for inherited fields that define picklists.
Can change?=No

REST: **isPicklistSuggested**[2] WIFE:

Boolean

Indicates whether the field allows users to enter their own values for a picklist. The value is set to *True* when a custom field is defined for Azure DevOps, Picklist (String), or Picklist (Integer) type is selected, and the checkbox for **Allow users to set their own values** is checked.
Can change?=Yes

REST: **isQueryable**
WIFE: **IsQueryable**

boolean

Indicates if the field shows up within the set of fields you can add to filter a work item query (True), or not (False). Most fields are queryable.

Can change?=No

REST:
WIFE: **IsReportable** [3]

boolean

Indicates if the reportable attribute is defined or set to anything other than **None**. This attribute can be changed for on-premises environments.
Can change?=Yes

REST:
WIFE: **IsUsedInGlobalWorkflow**

boolean

Indicates if the field is defined within a global workflow.
Can change?=No

REST:
WIFE: **IsUserNameField**

boolean

Indicates if the field is used to display an Identity field.
Can change?=No

REST: **name**
WIFE: **Name**

string

Friendly name assigned to the field. The friendly name can't be changed for Azure DevOps, but can be changed for on-premises using the **witadmin changefield** command.
Can change?=On-premises only

REST: **picklistId**
WIFE: **HelpText**

GUID

If the field is a picklist, the identifier of the associated picklist, otherwise null. A unique GUID value is assigned when a custom field is defined for Azure DevOps and Picklist

(String) or Picklist (Integer) type is selected.
Can change?=No

REST:
WIFE: **ProhibitedValues**

collection

Gets the collection of prohibited values for a field that specifies such values. You can only define prohibited values for on-premises deployments.
Can change?=On-premises only

REST: **readOnly**
WIFE:

Boolean

Indicates whether the field is set to read only. For Azure DevOps Services, only custom fields can be changed to be read-only. System fields can't be modified.
Can change?=Yes

REST: **referenceName**
WIFE: **ReferenceName**

string

Specifies the reference name of a field.
Can change?=No

REST:
WIFE: **ReportingAttributes**[3]

Specifies **Detail**, **Dimension**, or **Measure**, depending on whether and how you want the field to be included in reports. Data from fields that have a value other than **None** for this attribute are exported to the data warehouse and can be included in SQL reports.
Can change?=On-premises only

REST:
WIFE: **ReportingName**[3]

string

Specifies the label for a field when data appears in SQL reports. If you don't specify a value, the field's friendly name is used.
Can change?=On-premises only

REST:
WIFE: **ReportingReferenceName**[3]

string

Specifies a different reference name to a field that is used when data is exported to the relational data warehouse. If you don't specify a value, the fields reference name is used.
Can change?=On-premises only

REST: **supportedOperations**
WIFE:

set

The set of query operators that are valid for use when referencing this field. For a quick reference of supported operations based on data type, see Query quick reference, Operators, and macros supported for each data type.
Can change?=No

REST:
WIFE: **SupportsTextQuery**

boolean

Indicates whether the field supports text queries such as **Contains Words**, **Does Not Contains Words**.
Can change?=No

REST:
WIFE: **SystemType**

string

Specifies the data type of the field, referencing the system name such as System.DateTime or System.String.
Can change?=No

REST: **type**
WIFE: **FieldType**

string

Specifies the data type of the field, such as *Boolean*, *DateTime*, *Integer*, *String*, and so on. For a complete list and descriptions, see Query fields, operators, and macros.
Can change?=No

REST: **usage**
WIFE: **Usage**

string

Specifies whether the field is intended for use with work items (*WorkItem*) or work item link (*WorkItemLink*) objects. The usage for most fields is WorkItem. For a complete list of usage values, see Get Fields, FieldUsage.
Can change?=No

> **Note**
>
> 1. For on-premises deployments, you can enable indexing for a field to improve query response times when filtering on the field. For more information, see Indexed fields later in this article.
> 2. The **isPicklist** and **isPicklistSuggested** attributes are only assigned to custom fields defined for an inherited process. The Inherited process model is supported for Azure DevOps Server 2019 and later versions. For more information, see Inherited process model.
> 3. All reporting attributes are valid only for on-premises deployments whose projects have been configured to support SQL Server Reporting and SQL Server Analysis Services.

List field attributes

You can list the attributes assigned to a field by using the **Fields - Get REST API**. Replace *OrganizationName* with your actual organization name.

```
REST
```

```
https://dev.azure.com/OrganizationName/_apis/wit/fields/FieldReferenceName
```

For example, to list the attributes for the Iteration Path, use the reference name `System.IterationPath` for the `fabrikam` organization:

REST

```
https://dev.azure.com/fabrikam/_apis/wit/fields/System.IterationPath

**Returned data:**

> [!div class="tabbedCodeSnippets"]
> ```JSON
> {
> "name": "Iteration Path",
> "referenceName": "System.IterationPath",
> "description": "The iteration within which this bug will be fixed",
> "type": "treePath",
> "usage": "workItem",
> "readOnly": false,
> "canSortBy": true,
> "isQueryable": true,
> "supportedOperations": [
> {
> "referenceName": "SupportedOperations.Under",
> "name": "Under"
> },
> {
> "referenceName": "SupportedOperations.NotUnder",
> "name": "Not Under"
> },
> {
> "referenceName": "SupportedOperations.Equals",
> "name": "="
> },
> {
> "referenceName": "SupportedOperations.NotEquals",
> "name": "<>"
> },
> {
> "referenceName": "SupportedOperations.In",
> "name": "In"
> },
> {
> "name": "Not In"
> }
> ],
> "isIdentity": false,
> "isPicklist": false,
> "isPicklistSuggested": false,
> "url": "https://dev.azure.com/mseng/_apis/wit/fields/System.IterationPath"
> }
```

```
> ```
```

Add and modify fields

To add fields to a process, you add them to one or more work item types. For more information, see Customize an inheritance process.

Related articles

- Refer to the Query quick reference to quickly understand query syntax and usage.
- Explore the Work item field index to view all available work item fields.
- Add and manage fields for an inherited process to customize fields according to your inherited process requirements.
- Access the metadata reference for Azure Boards Analytics to understand analytics metadata and reporting capabilities.

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Feedback

Was this page helpful? Yes No

Provide product feedback

Rules and rule evaluation

Article • 03/25/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Rules are used to set or restrict value assignments to a work item field. There are two main types of rules, auto-generated rules and custom rules defined for a process or project. Auto-generated rules minimize the need to add custom rules for areas that should work in a standard way.

You define custom rules in order to support your business use cases. Depending on a field's data type, you can set various restrictions on what data can be entered into that field. You can specify values for a pick list (drop-down menu), set default values, clear entries, or restrict changes. With conditional rules, you can apply rules to a field based on dependencies between different fields' values. You can also restrict who can modify a field or scope a rule to only apply to a group.

Read this article to understand the following:

- How the system applies auto-generated rules
- Restrictions placed on definition of custom rules on system fields
- The different types of custom rules you can apply
- How rules are evaluated
- Difference between rules defined for an Inheritance process versus an On-premises XML process
- Why you should minimize the number of custom rules you define

Prior to defining custom rules, read Configure and customize Azure Boards to gain a broad understanding of how to customize Azure Boards to meet your business needs.

> **Tip**
>
> Minimize the number of rules you define for a WIT. While you can create multiple rules for a WIT, addition rules can negatively impact performance when a user adds and modifies work items. When users save work items, the system validates all rules associated with the fields for its work item type. Under certain conditions, the rule validation expression is too complex for SQL to evaluate.

Auto-generated rules

Auto-generated rules minimize the need to add custom rules for areas that should work in a standard way.

State transition rules

Inherited processes generate the entire set of any-to-any state transition rules dynamically for each custom work item type and custom state added to a workflow. A transition from any state to any state is valid.

For On-premises XML processes, you must specify the valid transitions within the `WORKFLOW` section of the work item type definition.

State transitions and By/Date field rules

By/Date fields correspond to **Created By/Date**, **Activated By/Date**, **Resolved By/Date**, and **Closed By/Date**.

For inherited processes, these fields are automatically set or cleared when you transition a work item from one state to another. The Changed By/Date fields aren't included as they are updated with each work item save and are unrelated to state transitions.

Default rules and behaviors that govern these fields include:

1. The *Closed* state is always contained in the *Completed* state category.
2. The *Completed* state category is not configurable, and is associated with one and only one State.
3. This *Closed* state is always *Closed* for Agile and CMMI processes, and always *Done* for Scrum and Basic processes.
4. Auto-generation of these rules are affected by locale as the rule condition contains the State name, which is localized. The system generates different rules for different locales.
5. Auto-generated rules for these fields are only specified for work item types that include these fields. It is possible for a work item type to not include one or more of these fields.
6. These rules are needed when a work item type has custom states, or the work item type is a custom work item type.
7. These rules only apply to inherited processes; they are never generated for the Hosted XML or On-premises XML processes.

Workflow states are associated with state categories to support the workflow on Kanban boards. To learn more, see How workflow states and state categories are used in Backlogs and Boards.

State Change Date field rules

These rules are technically a lot simpler than Closed By/Closed Date rules because they are not dependent on any particular state. For any work item type, the same rules will always work. They need to be auto-generated because some OOB work item types do not contain the State Change Date field, so when the user adds this field to a custom work item type, these rules need to be auto-generated as well. The same principles for Closed By/Closed Date rules apply here as well.

Custom rules

All custom rules are optional. For an inherited process, you specify a rule which consists of a condition plus action. For an On-premises XML process, you specify rules for a field or within the workflow.

There isn't a one-to-one mapping between the two processes. In some cases, the XML element rule is defined within the **Edit field** dialog for the inherited process and not as a rule. Other XML elements, such as `FROZEN`, `MATCH`, `NOTSAMEAS`, aren't supported in the inherited process.

Note the following:

- Rules are always enforced, not only when you are interacting with the form but also when interfacing through other tools. For example, setting a field as read-only not only applies the rule on the work item form, but also through the API and Excel Azure DevOps Server Add-in.
- Inherited process entries specify conditions and actions to make a complete rule. XML elements don't make those distinctions.
- Field rules don't support assigning values that are the sum of two other fields or performing other- mathematical calculations. However, you may find a solution that fits your needs via the TFS Aggregator (Web Service) Marketplace extension. See also Rollup of work and other fields.
- You may find additional solutions to applying custom rules to fields using a Marketplace extensions, such as the Work item form control library extension.

Rule composition

Inheritance process

For an inherited process, each rule consists of two parts: Conditions and Actions. Conditions define the circumstances which must be met in order for the rule to be

applied. Actions define the operations to perform. For most rules, you can specify a maximum of two conditions and 10 actions per rule. All custom rules require all conditions to be met in order to be run.

As an example, you can make a field required based on the value assigned to the state and another field. For example:

`(Condition) When a work item State is` *Active*
`(Condition) And when the value of` *Value Area = Business*
`(Action) Then make required` *Story Points*

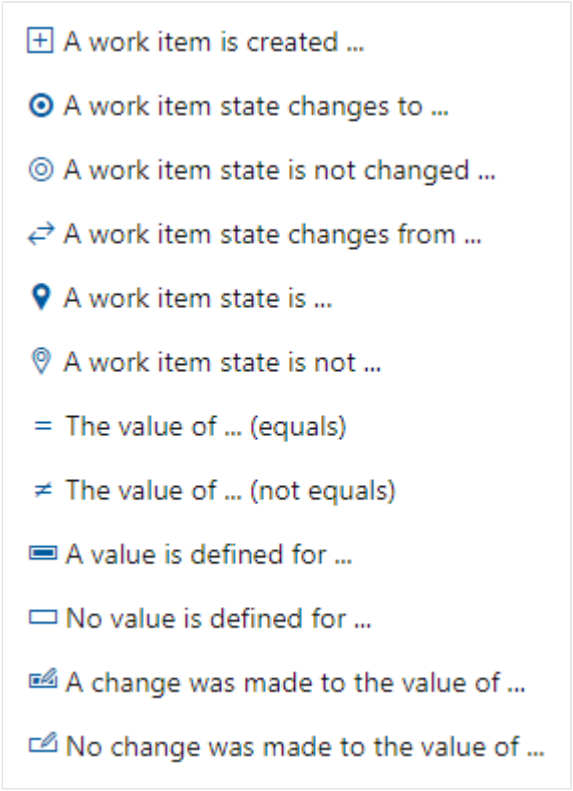
> **Note**
>
> Currently, only one condition is supported for state-transition rules. If you're applying rules based on State, see Apply rules to workflow states.

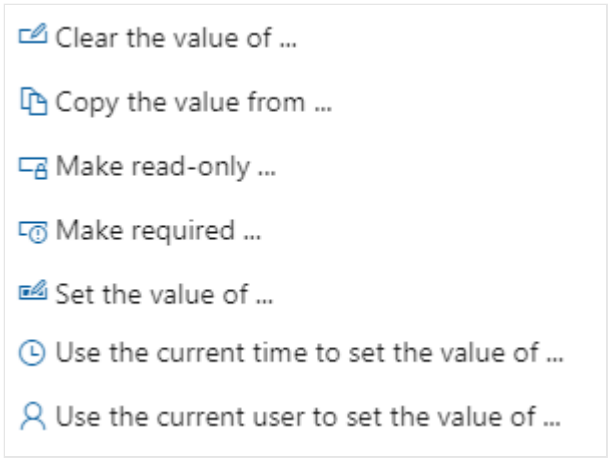
The following table summaries the Actions that are available with the selected Conditions.

Condition

Supported Actions

Set field value or make required or read-only

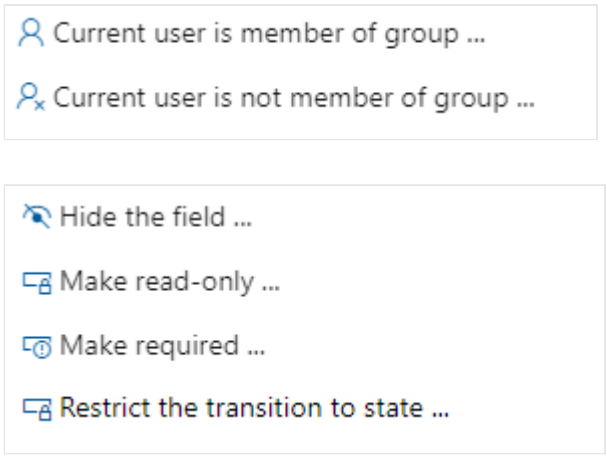




Restrict a transition based on State



Based on and user or group membership, set field attribute or restrict a State transition



What happens if too many rules are defined

A single SQL expression is defined per project to validate work items whenever they are created or updated. This expression grows with the number of rules you specify for all work item types defined for the project. Each behavioral qualifier specified for a field results in an increase in the number of sub-expressions. Nested rules, rules that apply only on a transition or conditioned on the value of some other field, cause more conditions to be added to an `IF` statement. Once the expression reaches a certain size

or complexity, SQL can't evaluate it any more and generates an error. Removing some WITs or eliminating some rules, can resolve the error.

You can specify values for a pick list (drop-down menu), set default values, clear entries, or restrict changes. With conditional rules, you can apply rules to a field based on dependencies between different fields' values. You can also restrict who can modify a field or scope a rule to only apply to a group.

Work item rules do not exist as a single collection. The rules are actually dynamically generated and merged from different data sources. The merge logic is a simple one, consolidate identical rules, but don't trim conflicting rules.

Bypass rules

In general, all work items are validated by the rule engine when users modify the work item. However, to support certain scenarios, users assigned the **Bypass rules on work item updates** project-level permission can save work items without rules being evaluated.

Rules can be bypassed in one of two ways. The first is through the Work Items - update REST API and setting the `bypassRules` parameter to `true`. The second is through the client object model, by initializing in bypassrules mode (initialize `WorkItemStore` with `WorkItemStoreFlags.BypassRules`).

System fields and custom rules

System fields have System.*Name* reference names, for example **System.Title** and **System.State**.

The following system fields are required to have a value: **Area ID**, **Changed Date**, **Created Date**, **Created By**, **State**, and **Reason**.

The rule engine restricts setting conditions or actions to system fields except as follows:

- You can make **State** and **Reason** fields read-only.
- You can apply most rules to the **Title**, **Assigned To**, **Description**, and **Changed By** fields.

If you don't see a field listed in the drop-down menu of the rule user interface for the Inheritance process, this is why. For example, if you try to make **Area Path** (System.AreaPath) read-only based on a condition, the Area Path field isn't available for

selection. Even if you're able to specify a system field, the rule engine may restrict you from saving the rule.

Default and copy rules

Default and copy rules modify the values of work item fields. They define run-time behavior and constraints, such as specifying default values, clearing fields, requiring fields to be defined, and more.

You can restrict application of these rules based on the current user's group membership as described in User or group membership rule restrictions.

Inheritance process

Most of these rule actions can be applied with the selection of any condition.

Inherited process action	Description
`Copy the value from...`	Specifies another field that contains a value to be copied into the current field.
`Clear the value of...`	Clears the field of any value that it contains.
`Use the current time to set the value of ...`	Sets the time for a field based on the current user's time setting.

Constraint rules

Constraint rules restrict changing the value of a field. They define the valid states for a work item. Each constraint operates on a single field. Constraints are evaluated on the server on work item save, and if any constraint is violated the save operation is rejected.

You can restrict application of these rules based on the current user's group membership as described in User or group membership rule restrictions.

Inheritance process

Most of these rule actions can be applied with the selection of any condition.

Inherited process action

Description

`Hide the field...`
Only available when a group membership condition is selected.

Specifies to not show the field on the work item form, essentially removing the ability for the current user to change the field's value.

`Make read-only`

Prevents a field from being modified at all. You might want to apply this rule under certain conditions. For example, after a work item is closed, you want to make a field read-only to preserve the data for reporting purposes.
To specify the field default is read-only, specify in Edit field dialog, **Options** tab.

`Make required`

Requires a user to specify a value for the field. Users cannot save a work item until they have assigned values to all required fields.
To specify the field default is required, specify in Edit field dialog, **Options** tab.

Pick lists

Pick lists define the values that a user can or can't choose for a String or Integer field. Values defined in a pick list appear on a work item form and the query editor.

Inheritance process

For an Inherited process, pick lists are defined through the Edit field dialog.

Edit field dialog

Description

Definition tab for a picklist field

Defines a list of allowed values for the field. Allowed values are values that are available for selection in a field list on work item forms and in the query builder. You must select from one of these values.

Check the **Allow users to enter their own values** checkbox within the **Options** tab to allow users to specify their own entries

Defines a list of suggested values for the field. Suggested values are values that are available for selection in a field list on work item forms and in the query builder. You can enter other values additionally to the ones in the list.

Conditional field values or changes

Conditional rules specify an action based on the value of a field equaling or not equaling a specific value, or if a change was or wasn't made to the value of a specific field. In general, conditional rules are applied first over unconditional rules. When multiple conditional rules evaluate to true, the order of execution is: When, WhenNot, WhenChanged, WhenNotChanged.

Inheritance process

You can specify multiple conditional rules per field. However, you can only specify a single driving field per conditional rule.

Inherited condition

Description

`The value of ... (equals)` [When]

Specifies one or more rules to apply to the current field when another field has a specific value.

`A change was made to the value of ...` [WhenChanged]

Applies one or more rules to the current field when a specific field's value is changed.

`The value of ... (not equals)` [WhenNot]

Applies one or more rules to the current field when another field does not have a specific value.

`No change was made to the value of ...` [WhenNotChanged]

Applies one or more rules to the current field when a specific field's value is not changed.

Inherited action

Description

`Clear the value of ...`
`Copy the value from ...`
`Make read-only ...`
`Make required ...`
`Set the value of ...`
`Use the current time to set the value of ...`
`Use the current user to set the value of ...`

Specifies the action to take on a specific field.

User or group membership rule restrictions

You can restrict application of a rule based on the current user's membership. We recommend you scope the rule to an Azure DevOps security group, and not a single user, although you can specify the latter. To have the rule scoped to multiple groups, you must create a parent Azure DevOps group that includes the set of groups that you want to use.

Process implementation

Tip

To avoid rule evaluation issues that may arise, specify Azure DevOps security groups and not Microsoft Entra ID or Active Directory security groups. To learn

more, see **Default rules and the rule engine**.

Inheritance process

As indicated in the following table, to restrict a rule based on the current user's membership, you specify one of two conditions for an Inherited process. These rules are active for Azure DevOps 2020 and later versions.

Applies to

Rule

Condition

`Current user is a member of group ...`

`Current user is not member of group ...`

Action

`Hide the field ...`

`Make read-only ...`

`Make required ...`

`Restrict the transition to state ...`

Use tokens to reference users or groups

Identity or people-picker fields can accept values that reference both users and groups. When you restrict a rule to a group, you indicate the domain or scope of the group. For some values, you can use tokens.

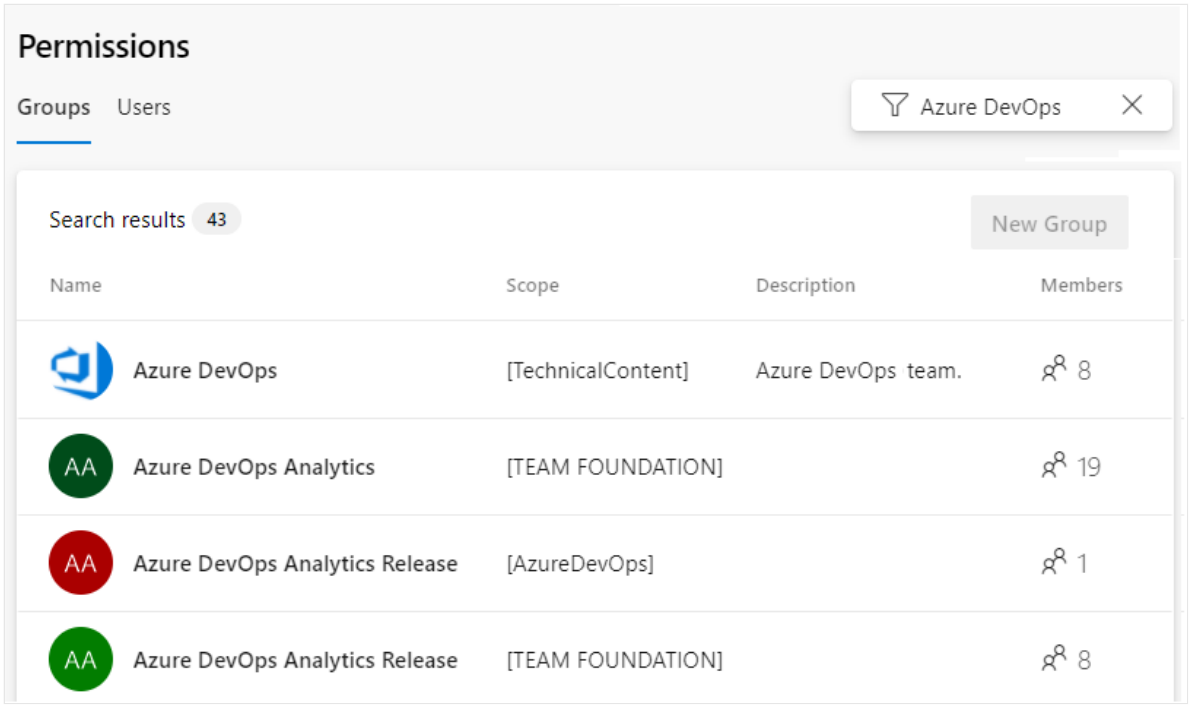
Inheritance process

Examples of tokens include the following:

- [*ProjectName*], such as [Fabrikam], [FabrikamFiber], [MyProject]
- [*OrganizationName*], such as [fabrikam], [myorganization]
- [*CollectionName*], such as [fabrikam], [myorganization]

To learn about the scopes available for your project or organization, go to the **Project Settings**>**Permissions**>**Groups** or **Organization**

Settings>Permissions>Groups page, you can filter the list as needed. For example, the following image shows the first four entries to a filtered list based on *Azure DevOps*. To learn more, see Change project-level permissions or Change project collection-level permissions.



To learn more about default security groups, see Permissions and groups

Rule evaluation

Rules that specify a condition based on the user or group membership of the user modifying a work item are evaluated in one of two ways. When the rule is evaluated, the application needs to determine whether the rule applies to the current user by checking if that user is or isn't a member of the specified group.

- When modifying the work item from the web portal, REST API, or **azure boards** command, a request to the Microsoft Entra ID or Active Directory is made. No problems occur for this operation.
- When modifying the work item from Visual Studio, Excel or other custom tool using the WIT Client Object Model, the request to evaluate membership is based on a client cache. The client cache is not aware of Active Directory groups.

> **Note**
>
> Visual Studio 2019 Team Explorer for projects using GIT has been re-written to use REST APIs.

To avoid problems with users updating work items from various clients, specify Azure DevOps security groups instead of Active Directory groups. You can easily create an Azure DevOps security group to correspond to an Active Directory group. To learn how, see Add or remove users or groups, manage security groups.

> **Note**
>
> The WIT Client OM is deprecated. As of January 1, 2020, it no longer is supported when working against Azure DevOps Services and Azure DevOps Server 2020.

Order in which rules are evaluated

Rules are typically processed in the sequence in which they are listed. However, the complete sequence for evaluation of all rules isn't fully deterministic.

This section describes the expected behavior and interactions when you apply conditional, copy, and default rules.

Inheritance process

The following steps show, in the correct sequence, the interactions that Azure DevOps performs and by the user of a work-item form. Only steps 1, 8, and 13 are performed by the user.

1. From an Azure DevOps client--such as the web portal or Visual Studio Team Explorer--a user creates a new work item or edits an existing work item.
2. Fill in field defaults. For all fields, apply any defaults assigned to the field that aren't part of a conditional clause.
3. Copy or set field values. For all fields, apply any rules to copy a value or set the value of a field that aren't part of a conditional clause.
4. For all fields with a When conditional rule that matches, apply rules to set or copy a field value.
5. For all fields with a When Not conditional rule that matches, apply rules to set or copy a field value.

   The system always processes **When** rules before **When Not** rules.
6. For all fields that have had their values changed since step 1 and that contain **When Changed** rules, apply rules to set or copy a field value.

7. Allow the user to start editing.
8. The user changes a field value and then moves focus from the field.
9. Process any **When** rules for that field that match the new value.
10. Process any **When Not** rules for that field that match the new value.
11. Process any **When Changed** rules for that field that match the new value.
12. Return editing ability to the user.
13. The user saves the changes to the data store.
14. For all fields, apply any `Use the current time to set the value of ...` actions that are defined for the field either directly or indirectly under a conditional rule.

Related articles

- Sample rule scenarios
- Work item field index
- Work item fields and attributes
- Change project-level permissions
- Change project collection-level permissions
- Permissions and groups
- **Inherited process**
  - Add a custom rule to a work item type
  - Apply rules to workflow states
- **On-premises XML process**
  - Define a default value or copy a value to a field
  - Define pick lists
  - Assign conditional-based values and rules

Feedback

Was this page helpful? Yes No

Provide product feedback

Add a work item type to a backlog and board

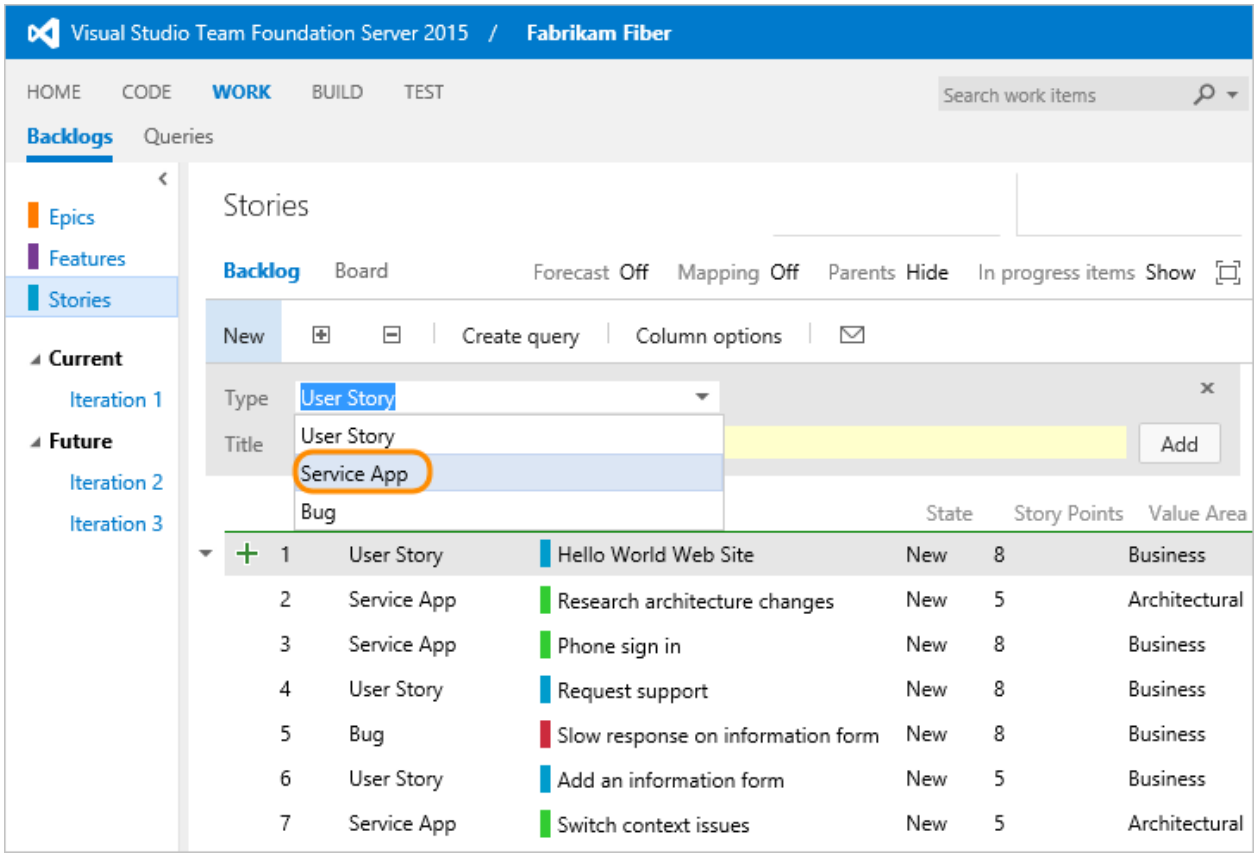
Article • 01/16/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

> **ⓘ Important**
>
> This article applies to project customization for Hosted XML and On-premises XML process models. For the Inheritance process model, see Customize your backlogs or boards for a process. For an overview of process models, see Customize your work tracking experience.

To satisfy select business processes, your team may want to track additional work item types (WITs) as requirements or as tasks. This way, you can add them and track them within the product and sprint backlogs or Kanban and task boards.

For example, you may want to track different types of requirements based on their target customer, or different types of bugs based on their source. Here the Fabrikam Fiber team has added the Service App WIT to track work that supports their customer service team.



> **Tip**
>
> If all you want to do is add bugs to your backlogs and boards, see **Show bugs on backlogs and boards** to make your team's selection. If you want to add another WIT and portfolio backlog, see **Add a portfolio backlog level**.

To add a WIT to track as requirements, tasks, or as bugs perform the corresponding steps:

- Add a WIT to track as requirements
- Add a WIT to track as tasks
- Add a WIT to the Bug Category

> **Note**
>
> We recommend that you add a WIT to one and only one of the following categories: Requirement Category, Task Category, or Bug Category. The system prevents you from adding the same WIT to both the Requirement Category and Task Category. WITs that you add to the Bug Category will follow the **bug behavior set for the team**.

Customization sequence

Hosted XML process model customization sequence

> **Note**
>
> Use the following guidance if you customize your process through import/export of definition files. Otherwise, if you customize your process through the admin UI, see **Add or modify a custom work item type for a process**.

You'll make your changes to your process definition files, and then import that process to either update existing projects or use the process to create a project.

- If you aren't the organization owner or a member of the Project Collection Administrator's group, get added.
- Export the process you want to update
- If you're new to customizing a process, read Customize a process for import.

On-premises XML process model customization sequence

You'll first export your work tracking definition files, update them, and then import them to your project.

- If you aren't a member of the Project Administrator or Project Collection Administrator's group, get added.
- Update your project to enable the latest features
- Export the following objects from your project:
  - WIT you want to add to backlogs and boards (if you haven't created it, do that now)
  - ProcessConfiguration
- If you're new to customizing work tracking objects, familiarize yourself with the following resources:
  - Add or modify a field
  - Add or modify a work item type
  - ProcessConfiguration XML reference.

Add a WIT to track it like a requirement

WITs that you add to the Requirement Category show up on the product backlog and Kanban board. You must make sure that the WIT definition contains required fields to support the Agile planning tools.

1. Export your process (Hosted XML) or your definition files (On-premises XML) as indicated in First steps.

2. Edit the WIT definition to support planning tools.

   Customize / Syntax to add or update

   Stack Rank field (Agile, CMMI)

   XML

   ```
   <FIELD name="Stack Rank" refname="Microsoft.VSTS.Common.StackRank"
   type="Double" reportable="dimension">
   <HELPTEXT>Work first on items with lower-valued stack rank. Set in
   triage.
   </HELPTEXT>
   </FIELD>
   ```

   Backlog Priority field (Scrum)

XML

```
<FIELD name="Backlog Priority"
refname="Microsoft.VSTS.Common.BacklogPriority" type="Double"
reportable="detail" />
```

Value Area field

XML

```
<FIELD name="Value Area" refname="Microsoft.VSTS.Common.ValueArea"
type="String">
<REQUIRED />
<ALLOWEDVALUES>
<LISTITEM value="Architectural" />
<LISTITEM value="Business"/>
</ALLOWEDVALUES>
<DEFAULT from="value" value="Business" />
<HELPTEXT>Business = delivers value to a user or another system;
Architectural = work to support other stories or components
</HELPTEXT>
</FIELD>
```

Requirement Type field (CMMI)

XML

```
<FIELD name="Requirement Type"
refname="Microsoft.VSTS.CMMI.RequirementType" type="String"
reportable="dimension">
<REQUIRED />
<ALLOWEDVALUES>
<LISTITEM value="Scenario" />
<LISTITEM value="Quality of Service" />
<LISTITEM value="Functional" />
<LISTITEM value="Operational" />
<LISTITEM value="Interface" />
<LISTITEM value="Security" />
<LISTITEM value="Safety" />
<LISTITEM value="Business Objective" />
<LISTITEM value="Feature" />
</ALLOWEDVALUES>
<DEFAULT from="value" value="Functional" />
</FIELD>
```

`WORKFLOW` section Update or verify the `WORKFLOW` section as follows:

- Agile, User Story: Add transitions from `Active` to `Removed` and `Resolved` to `Removed`; remove rules that populate `Activated By` and `Activated Date` fields

when state= `Resolved`

- Scrum, Product backlog item: Add transition from `Committed` to `Removed`
  If you've customized the `WORKFLOW`, make sure to define the required state-to-metastate mappings in the ProcessConfiguration RequirementBacklog section.

`FORM` section Add or verify the following fields have been added to the `FORM` section:

- **Agile:**

XML

```
<Control FieldName="Microsoft.VSTS.Scheduling.StoryPoints"
Type="FieldControl" Label="Story Points" LabelPosition="Left" />
<Control FieldName="Microsoft.VSTS.Common.ValueArea"
Type="FieldControl" Label="Value area" LabelPosition="Left" />
```

- **CMMI:**

XML

```
<Control Type="FieldControl" FieldName="Microsoft.VSTS.Scheduling.Size"
Label="Size" LabelPosition="Left" />
<Control Type="FieldControl"
FieldName="Microsoft.VSTS.CMMI.RequirementType" Label="Type"
LabelPosition="Left" />
<Control FieldName="Microsoft.VSTS.Common.ValueArea"
Type="FieldControl" Label="Value area" LabelPosition="Left" />
```

- **Scrum:**

XML

```
<Control FieldName="Microsoft.VSTS.Scheduling.Effort"
Type="FieldControl" Label="Effort" LabelPosition="Left" />
<Control FieldName="Microsoft.VSTS.Common.ValueArea"
Type="FieldControl" Label="Value area" LabelPosition="Left" />
```

3. Edit the Categories definition. Add the WIT to the Requirement category. Here we add Service App.

XML

```
<CATEGORY name="Requirement Category"
refname="Microsoft.RequirementCategory">
   <DEFAULTWORKITEMTYPE name="User Story" />
   <WORKITEMTYPE name="Service App" />
</CATEGORY>
```

4. Update or verify ProcessConfiguration definition: `RequirementBacklog` section for the WIT you're adding. Specifically, make sure that the following conditions are met:

   - Map the start of each workflow state to `type="Proposed"`
   - Map each intermediate workflow state you want to have show up on the Kanban board to `type="InProgress"`
   - Map the end of each workflow state to `type="Complete"`
   - Make sure that you have only one State mapped to `type="Complete"`

     For example, add the Pending workflow state:

   XML

   ```
   <States>
        <State value="New" type="Proposed" />
        <State value="Active" type="InProgress" />
        <State value="Pending" type="InProgress" />
        <State value="Resolved" type="InProgress" />
        <State value="Closed" type="Complete" />
   </States>
   ```

5. Add the WIT color definition to the ProcessConfiguration `WorkItemColors` section. For example:

   XML

   ```
   <WorkItemColor primary="FF33CC33" secondary="FFD6F5D6" name="Service
   App" />
   ```

6. Update your project:

   - **Hosted XML:** Import your process.
   - **On-premises XML:** Import the definition files you updated in this order:
     a. WIT b. Categories
     c. ProcessConfiguration

Add a WIT to track it like a task

WITs that you add to the Task Category show up on the sprint backlogs and task boards. The WIT you add must specify required fields to support the Agile planning tools.

1. Export your process (Hosted XML) or your definition files (On-premises XML) as indicated in First steps.

2. Edit the WIT definition to support planning tools.

Customize / Syntax to add or update

Stack Rank field(Agile, CMMI)

XML

```
<FIELD name="Stack Rank" refname="Microsoft.VSTS.Common.StackRank"
type="Double" reportable="dimension">
<HELPTEXT>Work first on items with lower-valued stack rank. Set in
triage.
</HELPTEXT>
</FIELD>
```

** Backlog Priority** field (Scrum)

XML

```
<FIELD name="Backlog Priority"
refname="Microsoft.VSTS.Common.BacklogPriority" type="Double"
reportable="detail" />
```

Activity field (Agile, Scrum)

XML

```
<FIELD name="Activity" refname="Microsoft.VSTS.Common.Activity"
type="String" reportable="dimension">
<HELPTEXT&gt;Type of work involved&lt/HELPTEXT>
<SUGGESTEDVALUES&gt;
<LISTITEM value="Development" />
<LISTITEM value="Testing" />
<LISTITEM value="Requirements" />
<LISTITEM value="Design" />
<LISTITEM value="Deployment" />
<LISTITEM value="Documentation" />
</SUGGESTEDVALUES>
</FIELD>
```

Discipline field (CMMI)

XML

```
<FIELD name="Discipline" refname="Microsoft.VSTS.Common.Discipline"
type="String" reportable="dimension">
<ALLOWEDVALUES&gt;
<LISTITEM value="Analysis" />
<LISTITEM value="User Experience" />
```

```
<LISTITEM value="User Education" />
<LISTITEM value="Development" />
<LISTITEM value="Test" />
</ALLOWEDVALUES>
</FIELD>
```

Remaining Work field (all)

XML

```
<FIELD name="Remaining Work"
refname="Microsoft.VSTS.Scheduling.RemainingWork" type="Double"
reportable="measure" formula="sum">
<HELPTEXT>
An estimate of the work remaining to complete the task (in person
hours)
</HELPTEXT>
</FIELD>
```

Additional scheduling fields (Agile, CMMI)

XML

```
<FIELD name="Original Estimate"
refname="Microsoft.VSTS.Scheduling.OriginalEstimate" type="Double"
reportable="measure" formula="sum">
<HELPTEXT&gt; The original estimate of work required to complete the
task (in person hours)</HELPTEXT>
</FIELD>
<FIELD name="Remaining Work"
refname="Microsoft.VSTS.Scheduling.RemainingWork" type="Double"
reportable="measure" formula="sum">
<HELPTEXT>An estimate of the work remaining to complete the task (in
person hours)
</HELPTEXT>
</FIELD>
<FIELD name="Completed Work"
refname="Microsoft.VSTS.Scheduling.CompletedWork" type="Double"
reportable="measure" formula="sum">
<HELPTEXT>
The work that has been completed for this task (in person hours)
<HELPTEXT>
</FIELD>
<FIELD name="Start Date" refname="Microsoft.VSTS.Scheduling.StartDate"
type="DateTime" reportable="dimension" />
<FIELD name="Finish Date"
refname="Microsoft.VSTS.Scheduling.FinishDate" type="DateTime"
reportable="dimension" />
```

`WORKFLOW` section

Update or verify the `WORKFLOW` section as follows:

- Agile, User Story: Add transitions from `Active` to `Removed` and `Resolved` to `Removed`; remove rules that populate `Activated By` and `Activated Date` fields when state= `Resolved`
- Scrum, Product backlog item: Add transition from `Committed` to `Removed`

  If you've customized the `WORKFLOW`, make sure to define the required state-to-metastate mappings in the ProcessConfiguration TestBacklog section.

`FORM` section

Add or verify the following fields have been added to the `FORM` section:

- **Agile:**

```
XML

<Control FieldName="Microsoft.VSTS.Scheduling.StoryPoints"
Type="FieldControl" Label="Story Points" LabelPosition="Left" />
<Control FieldName="Microsoft.VSTS.Common.ValueArea"
Type="FieldControl" Label="Value area" LabelPosition="Left" />
```

- **CMMI:**

```
XML

<Control Type="FieldControl" FieldName="Microsoft.VSTS.Scheduling.Size"
Label="Size" LabelPosition="Left" /&gt;
<Control FieldName="Microsoft.VSTS.Common.ValueArea"
Type="FieldControl" Label="Value area" LabelPosition="Left" />
```

- **Scrum:**

```
XML

<Control FieldName="Microsoft.VSTS.Scheduling.Effort"
Type="FieldControl" Label="Effort" LabelPosition="Left" />
<Control FieldName="Microsoft.VSTS.Common.ValueArea"
Type="FieldControl" Label="Value area" LabelPosition="Left" />
```

3. Edit the Categories definition. Add the WIT to the Task category. Here we add Service Task.

```
XML

<CATEGORY name="Task Category" refname="Microsoft.TaskCategory">
   <DEFAULTWORKITEMTYPE name="Task" />
```

```
    <WORKITEMTYPE name="Service Task" />
</CATEGORY>
```

4. Update or verify ProcessConfiguration definition: `TaskBacklog` section for the WIT you're adding.
   Specifically, make sure that the following conditions are met:

   - Map the start of each workflow state to `type="Proposed"`
   - Map each intermediate workflow state you want to have show up on the task board to `type="InProgress"`
   - Map the end of each workflow state to `type="Complete"`
   - Make sure that you have only one State mapped to `type="Complete"`

   For example, add the `Blocked` workflow state:

   XML

   ```
   <States>
      <State value="New" type="Proposed" />
      <State value="Active" type="InProgress" />
      <State value="Blocked" type="InProgress" />
      <State value="Resolved" type="InProgress" />
      <State value="Closed" type="Complete" />
   </States>
   ```

5. Add the WIT color definition to the ProcessConfiguration `WorkItemColors` section.
   For example:

   XML

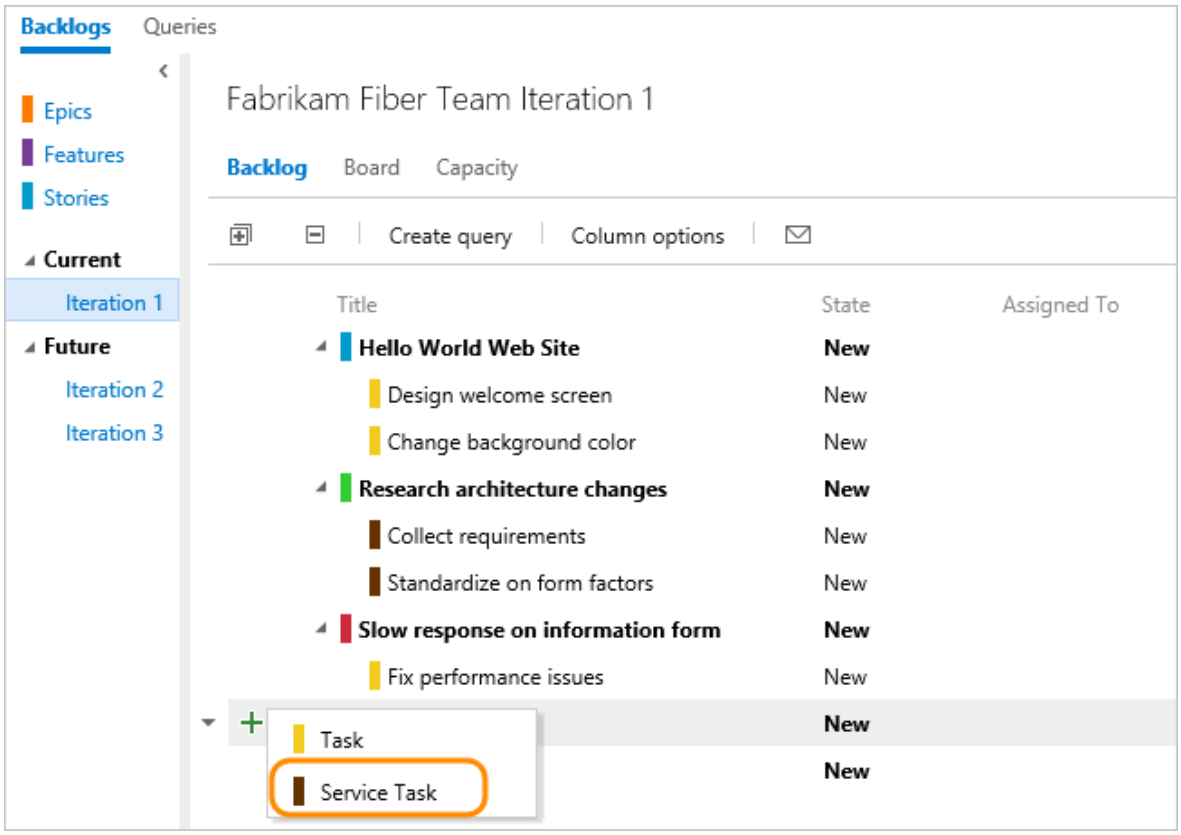
   ```
   <WorkItemColor primary="FFF2CB1D" secondary="FFF6F5D2" name="Service
   Task" />
   ```

6. Update your project:

   - **Hosted XML:** Import your process.
   - **On-premises XML:** Import the definition files you updated in this order:
     a. WIT
     b. Categories
     c. ProcessConfiguration

7. Confirm that you can add the WIT to the task board. Open the task board page, or refresh the page if it's already open.
   You should be able to select either Task or Service Task as a linked work item to a

user story.



On the task board, drag the work item to update its status. You'll notice that you can't drag an item to a column for which the state isn't valid. For example, you can't drag Task to the Blocked state, or a Task into a resolved state. If you want to use these states, then add them to the workflow definition for the corresponding WIT definition.

Add a WIT to the Bug Category

WITs that you add to the Bug Category will be treated based on the team setting. Because these WITs may be treated either as requirements or tasks, they must meet Agile planning tool requirements for both requirements and tasks.

1. Export your process (Hosted XML) or your definition files (On-premises XML) as indicated in First steps.

2. Edit the WIT definition to support planning tools by meeting the conditions listed for Step 2 of Add a WIT to track as requirements and Add a WIT to track as tasks.

3. Edit the Categories definition. Add the WIT to the Bug Category. Here we add two WITs.

```
XML
```

```
<CATEGORY name="Bug Category" refname="Microsoft.BugCategory">
   <DEFAULTWORKITEMTYPE name="Bug" />
   <WORKITEMTYPE name="Service Bug" />
   <WORKITEMTYPE name="Feedback" />
</CATEGORY>
```

4. Update or verify ProcessConfiguration definition: `BugWorkItems` section for the WIT you're adding.

   Specifically, make sure that the following conditions are met:

   - Map the start of each workflow state to `type="Proposed"`
   - Map each intermediate workflow state you want to have show up on the Kanban or task board to `type="InProgress"`
   - Map the end of each workflow state to `type="Complete"`
   - Make sure that you have only one State mapped to `type="Complete"`

   For example, add the `Investigate` workflow state:

   XML

   ```
   <States>
    <State value="New" type="Proposed" />
    <State value="Active" type="InProgress" />
    <State value="Investigate" type="InProgress" />
    <State value="Resolved" type="InProgress" />
    <State value="Closed" type="Complete" />
   </States>
   ```

5. Add the WIT color definition to the ProcessConfiguration `WorkItemColors` section. For example:

   XML

   ```
   <WorkItemColor primary="FFF2CB1D" secondary="FFF6F5D2" name="Service
   Bug" />
   <WorkItemColor primary="FFFF00FF" secondary="FFFFCCFF" name="Feedback"
   />
   ```

6. Update your project:

   - **Hosted XML:** Import your process.
   - **On-premises XML:** Import the definition files you updated in this order:
     a. WIT

b. Categories
c. ProcessConfiguration

7. Choose your team setting for how bugs are tracked.

8. Confirm that the WIT appears on the backlogs and boards as expected and that you can add it through the backlogs as expected.

Import and export definition files (On-premises XML)

Use the **witadmin** commands to import and export definition files. For details, see witAdmin: Customize and manage objects for tracking work.

> **Tip**
>
> With **witadmin**, you can **import and export definition files**. Other tools you can use include the Process Editor (requires that you have installed a version of Visual Studio). **Install the Process Template editor from the Visual Studio Marketplace**.

1. Open a Command Prompt window according to the instructions provided in witAdmin: Customize and manage objects for tracking work. For example:

```
%programfiles(x86)%\Microsoft
```

1. Enter the `witadmin` command, substituting your data for the arguments that are shown. For example, to import a WIT:

```
witadmin importwitd /collection:CollectionURL /p:"ProjectName"
/f:"DirectoryPath\WITDefinitionFile.xml"
```

For *CollectionURL* specify the URL of a project collection and for *ProjectName* specify the name of a project defined within the collection. You must specify the URL in the following format: `http://ServerName:Port/VirtualDirectoryName/CollectionName`.

For *DirectoryPath*, specify the path to the `WorkItem Tracking/TypeDefinitions` folder that holds the process template that you downloaded. The directory path

must follow this structure: `Drive:\TemplateFolder\WorkItem Tracking\TypeDefinitions`.

For example, import the ServiceApp WIT:

```
witadmin importwitd
/collection:"http://MyServer:8080/tfs/DefaultCollection"/p:MyProject
/f:"DirectoryPath/ServiceApp.xml"
```

Use these commands to export and import WITs, categories, and process configuration:

```
witadmin exportwitd /collection:CollectionURL /p:"ProjectName" /n:TypeName
/f:"DirectoryPath\WITDefinitionFile.xml"
witadmin importwitd /collection:CollectionURL /p:"ProjectName"
/f:"DirectoryPath\WITDefinitionFile.xml"
witadmin exportcategories /collection:"CollectionURL" /p:"ProjectName"
/f:"DirectoryPath/categories.xml"
witadmin importcategories /collection:"CollectionURL" /p:"ProjectName"
/f:"DirectoryPath/categories.xml"
witadmin exportprocessconfig /collection:"CollectionURL" /p:"ProjectName"
/f:"DirectoryPath/ProcessConfiguration.xml"
witadmin importprocessconfig /collection:"CollectionURL" /p:"ProjectName"
/f:"DirectoryPath/ProcessConfiguration.xml"
```

Related articles

We've just shown how to add another WIT to your backlogs or boards. However, if you want to add another WIT to act as a portfolio backlog, see Add portfolio backlogs.

See Customize your work tracking experience for an overview of all the options available for customizing work tracking objects.

To learn how hierarchical or nested elements are displayed on a backlog or board, see How backlogs and boards display hierarchical (nested) items.

Feedback

Was this page helpful? Yes No

Provide product feedback

Add a portfolio backlog level

Article • 09/22/2023

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

Your project comes equipped with two portfolio backlogs: Features and Epics. However, if you need one or more additional portfolio backlogs, you can add them.

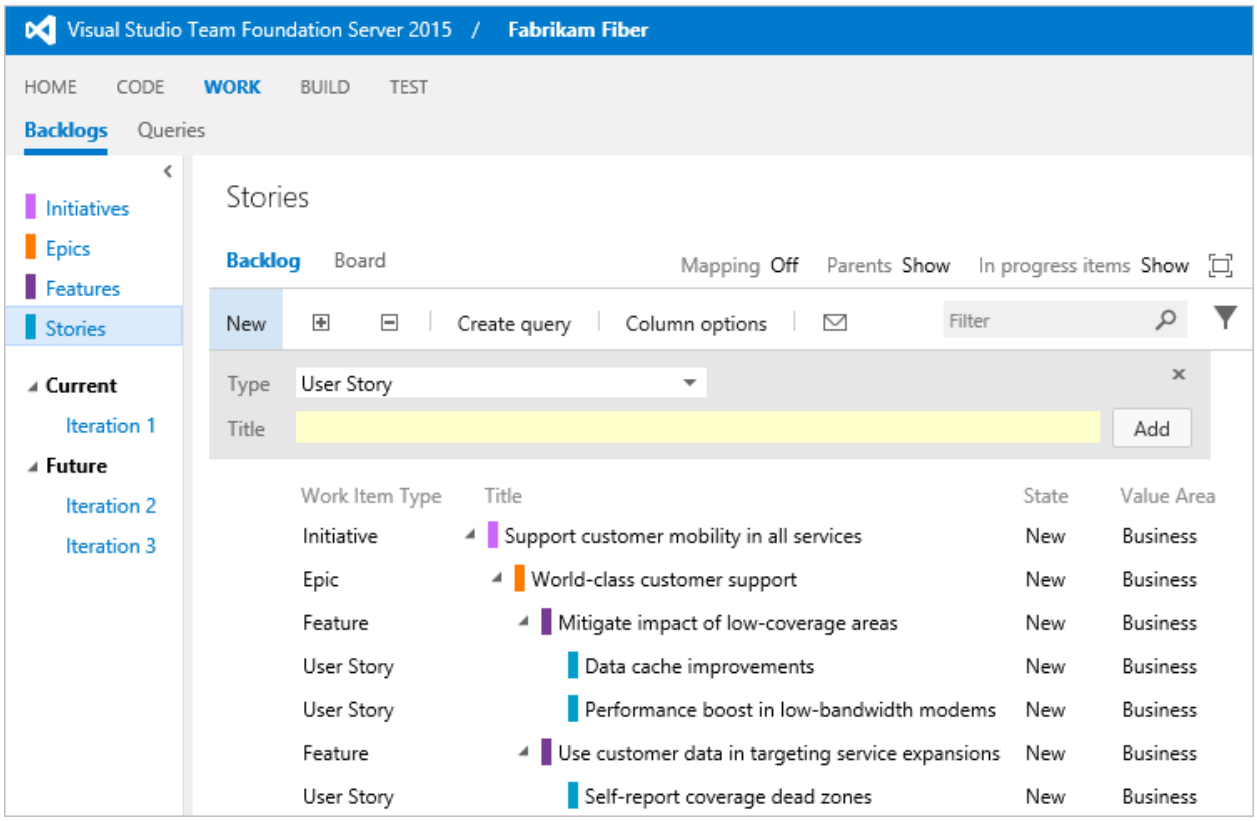
> **Important**
>
> This article applies to project customization for Hosted XML and On-premises XML process models. For the Inheritance process model, see **Customize your backlogs or boards for a process**.
>
> For an overview of process models, see **Customize your work tracking experience**.

Portfolio backlogs are useful for organizing your backlog under business initiatives. When you organize your backlogs into portfolios, you can gain a hierarchical view of the work defined in lower-level backlogs, including work in progress across several teams. Program managers can track the status of those backlog items of interest and drill down to ensure that all work is represented.

> **Note**
>
> If you haven't yet enabled the Portfolio Backlogs feature for your on-premises Azure DevOps project, **do that first**.

Here, we add a third portfolio backlog, Initiative. With it, the management team can set priorities and view progress of work belonging to initiatives.

> **Note**
>
> The images you see from your web portal may differ from the images you see in this topic. These differences result from updates made to your project, and the process template chosen when creating your project—**Agile**, **Scrum**, or **CMMI**. However, the basic functionality available to you remains the same unless explicitly mentioned.



You can add up to five levels of portfolio backlogs. And, each team can choose which backlogs appear for them to work on.

Process overview

The process to add another portfolio backlog differs slightly depending on the process model you use.

- For **Hosted XML**: You'll first export your process, add or update definition files, and then import that process to either update existing projects or use it to create a project.
- For **On-premises XML**: You'll first export your work tracking definition files, update them, and then import them to your project.

This article walks you through adding a portfolio backlog to a project based on the Agile process in these five steps:

1. Export the files you need
2. Create the Initiative work item type
3. Update Categories with the Initiative Category
4. Update ProcessConfiguration to add the Initiative portfolio backlog
5. Update your project and verify your changes

You can apply the same steps if you work with a project based on the Scrum or CMMI process. When you're done, you'll get to manage your portfolio of projects by grouping

work within these four levels: User Stories (or Product backlog items or Requirements), Features, Epics, and Initiatives.

For more informaiton, see About processes and process templates. For an overview of the three process models, see Customize your work tracking experience.

1. Export the files you need

1. If you aren't a member of the **Project Collection Administrators** group, get added as an administrator. You need these permissions to customize the project.

2. Get the files you need:

    - For **Hosted XML**: Export the process you want to update. Save the files to a folder that you'll use to update these files and folders: Categories, ProcessConfiguration, and WorkItemTypes.
    - For **On-premises XML**: Export the definition files you'll need: Epic, Categories, and ProcessConfiguration.

2. Create a work item type named Initiative

The easiest way to create a work item type (WIT) is to copy an existing one, rename it, and edit it to support your requirements. In this example, we copy the Epic WIT and label it Initiative.

- Copy the `Epic` WIT definition to an XML file labeled `Initiative`. (The Epic.xml file is located in the WorkItem Tracking folder of the ProcessTemplate folder.)

    - Edit the file named `Initiative`.

- Rename the WIT. Replace `WORKITEMTYPE name="Epic"` with `WORKITEMTYPE name="Initiative"`, and update the description.
`<WORKITEMTYPE name="Initiative" >`
  `<DESCRIPTION>Initiatives help program managers to effectively manage and organize work across several teams >`
  `</DESCRIPTION>`
`...`
`</WORKITEMTYPE>`

- Add any custom fields that you want to track using this WIT.
- Rename the `Tab` section named `Features` to `Epics` and replace `Filter WorkItemType="Feature"` with `Filter WorkItemType="Epic"`.

`<Tab Label="Epics">`
`<Control Type="LinksControl" Name="Hierarchy">`
`<LinksControlOptions>`
`<WorkItemLinkFilters FilterType="include">`
`<Filter LinkType="System.LinkTypes.Hierarchy" />`
`</WorkItemLinkFilters>`
`<WorkItemTypeFilters FilterType="include">`
`<Filter WorkItemType="Epic" />`
`</WorkItemTypeFilters>`
`<ExternalLinkFilters FilterType="excludeAll" />`
`<LinkColumns>`
`<LinkColumn RefName="System.ID" />`
`<LinkColumn RefName="System.Title" />`
`<LinkColumn RefName="System.AssignedTo" />`
`<LinkColumn RefName="System.State" />`
`<LinkColumn LinkAttribute="System.Links.Comment" />`
`</LinkColumns>`
`</LinksControlOptions>`
`</Control>`
`</Tab>`

With this change, you cause the tab control to exclusively show or link to epics as child work items of the initiative.

3. Update Categories with the Initiative Category

Now, add the Initiative Category. This adds the Initiative backlog to process configuration. The agile experience manages WITs according to categories.

Add the Initiative Category to the Categories.xml file. (The Categories.xml file is located in the WorkItem Tracking folder.)

XML

```
  <CATEGORY name="Initiative Category"
refname="FabrikamFiber.InitiativeCategory">
    <DEFAULTWORKITEMTYPE name="Initiative" />
  </CATEGORY>
```

You can add this category anywhere within the definition file. Since you are adding a custom category, label the category using your company name.

4. Update ProcessConfiguration to add the Initiative portfolio backlog

In this last step, you add the Initiative portfolio backlog to the process and modify the Feature portfolio backlog to reflect the hierarchy between Initiatives and Features. The process configuration determines the parent-child relationships among the portfolio backlogs.

1. Edit the ProcessConfiguration file to add a new portfolio backlog within the `PortfolioBacklogs` section. (The ProcessConfiguration.xml file is located in the WorkItem Tracking/Process folder of the ProcessTemplate folder.)

   Add the Initiative Category by adding the following syntax. Replace the names, workflow state values, and default column fields to match those that you use.

   XML

   ```
   <PortfolioBacklog category="FabrikamFiber.InitiativeCategory"
   pluralName="Initiatives" singularName="Initiative"
   workItemCountLimit="1000">
     <States>
       <State value="New" type="Proposed" />
       <State value="Active" type="InProgress" />
       <State value="Resolved" type="InProgress" />
       <State value="Closed" type="Complete" />
     </States>
     <Columns>
       <Column refname="System.WorkItemType" width="100" />
       <Column refname="System.Title" width="400" />
       <Column refname="System.State" width="100" />
       <Column refname="Microsoft.VSTS.Scheduling.Effort" width="50" />
       <Column refname="Microsoft.VSTS.Common.BusinessValue" width="50" />
       <Column refname="Microsoft.VSTS.Common.ValueArea" width="100" />
       <Column refname="System.Tags" width="200" />
     </Columns>
     <AddPanel>
       <Fields>
         <Field refname="System.Title" />
       </Fields>
     </AddPanel>
   </PortfolioBacklog>
   ```

   If you have modified the workflow states, then verify that each work flow state is mapped to one of the metastates of `Proposed`, `InProgress`, and `Complete`. The last

state within the workflow must map to `Complete`.

2. Edit the `PortfolioBacklog` element for the Epic Category to point to `Initiative` as the parent backlog.

```
XML

<PortfolioBacklog category="Microsoft.EpicCategory" pluralName="Epics"
   singularName="Epic" parent="FabrikamFiber.InitiativeCategory"
   workItemCountLimit="1000">
   . . .
</PortfolioBacklog>
```

   Intermediate portfolio backlogs require specifying the parent category, which must be configured as a portfolio backlog.

3. Add the color to use for Initiative to the `WorkItemColors` section.

```
XML

    <WorkItemColor primary="FFCC66FF" secondary="FFF0D1FF"
name="Initiative" />
```

   This assigns a bright pink as the primary color to use in list displays, and a paler pink for the secondary color (currently not used).

5. Update your project and verify access to the new portfolio backlog

1. Update your project:

   - For **Hosted XML:** Import your process.
   - For **On-premises XML:** Import the definition files you updated in this order:
     a. Initiative.xml
     b. Categories.xml
     c. ProcessConfiguration.xml

2. Open or refresh the web portal and confirm that Initiative appears as a portfolio backlog as expected. See Organize your backlog.

3. Grant Advanced access to users who'll need to exercises all the features available with portfolio backlogs.
   For **Hosted XML:** See Assign licenses to users.

Import and export definition files (on-premises TFS only)

If you're updating a project that connects to an on-premises TFS, you'll use the **witadmin** commands to import and export definition files. You need to export the following files:

- Epic.xml
- Categories.xml (located in the WorkItem Tracking folder)
- ProcessConfiguration.xml (located in the WorkItem Tracking/Process folder)

> **Tip**
>
> With **witadmin**, you can **import and export definition files**. Other tools you can use include the Process Editor (requires that you have installed a version of Visual Studio). **Install the Process Template editor from the Visual Studio Marketplace**.

1. Open a Command Prompt window according to the instructions provided in witAdmin: Customize and manage objects for tracking work. For example:

```
%programfiles(x86)%\Microsoft
```

1. Enter the `witadmin` command, substituting your data for the arguments that are shown. For example, to import a WIT:

```
witadmin importwitd /collection:CollectionURL /p:"ProjectName"
/f:"DirectoryPath\WITDefinitionFile.xml"
```

   For *CollectionURL* specify the URL of a project collection and for *ProjectName* specify the name of a project defined within the collection. You must specify the URL in the following format: `http://ServerName:Port/VirtualDirectoryName/CollectionName`.

   For *DirectoryPath*, specify the path to the `WorkItem Tracking/TypeDefinitions` folder that holds the process template that you downloaded. The directory path must follow this structure: `Drive:\TemplateFolder\WorkItem Tracking\TypeDefinitions`.

For example, import the ServiceApp WIT:

```
witadmin importwitd
/collection:"http://MyServer:8080/tfs/DefaultCollection"/p:MyProject
/f:"DirectoryPath/ServiceApp.xml"
```

Use these commands to export and import categories and process configuration:

```
witadmin exportwitd /collection:CollectionURL /p:"ProjectName" /n:TypeName
/f:"DirectoryPath\WITDefinitionFile.xml"

witadmin importwitd /collection:CollectionURL /p:"ProjectName"
/f:"DirectoryPath\WITDefinitionFile.xml"

witadmin exportcategories /collection:"CollectionURL" /p:"ProjectName"
/f:"DirectoryPath/categories.xml"

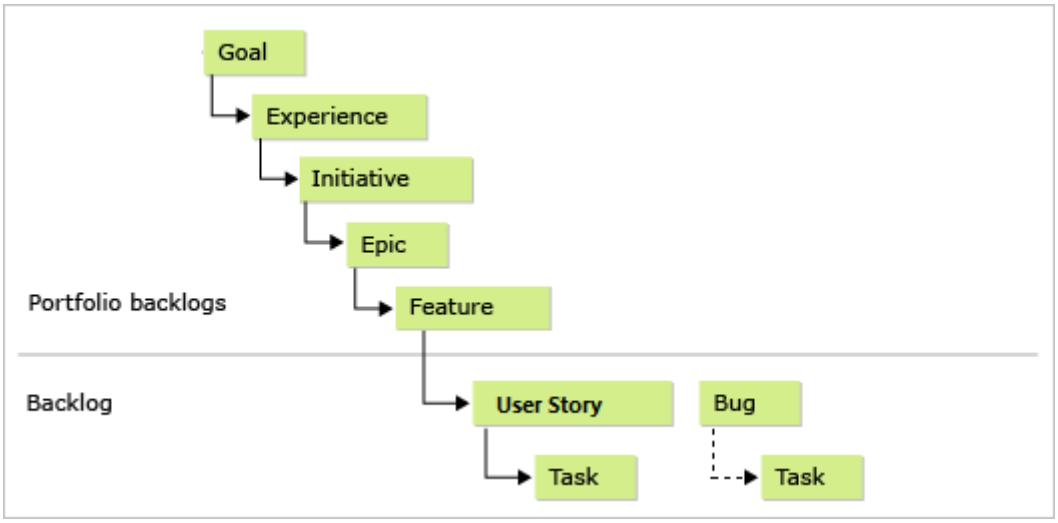
witadmin importcategories /collection:"CollectionURL" /p:"ProjectName"
/f:"DirectoryPath/categories.xml"

witadmin exportprocessconfig /collection:"CollectionURL" /p:"ProjectName"
/f:"DirectoryPath/ProcessConfiguration.xml"

witadmin importprocessconfig /collection:"CollectionURL" /p:"ProjectName"
/f:"DirectoryPath/ProcessConfiguration.xml"
```

Related articles

We've just shown how to add another portfolio backlog level. You can add up to five portfolio backlogs. This includes the default backlogs of Feature and Epic. In total, this provides you with seven levels from the top-level portfolio backlog to task.



If you want to add another WIT to your backlogs or boards, see work item types to backlogs and boards.

To learn more about the syntax for a definition file or command line tool, see these topics:

- All WITD XML elements reference
- Categories XML element reference
- Process configuration XML element reference
- Import, export, and manage work item types
- Import and export categories
- Import and export process configuration

Otherwise, see Customize your work tracking experience to access other configuration and customization options available to you.

If you have additional questions, see Team Foundation Server - work tracking forum.

Portfolio backlog hierarchy

What controls the hierarchy among portfolio backlogs?

The process configuration determines the hierarchy through the assignment of parent categories to portfolio backlog categories. Only parent-child relationships are supported. The upper-most category within the hierarchy doesn't contain a parent assignment.

Portfolio backlogs and WIT categories

Can I have more than one WIT defined in a category that I use for a portfolio backlog?

Yes. For example, you can add Goal and Initiative WITs to a portfolio backlog category. The main restriction is to not add the same WIT to two different categories that are assigned to one of the following sections for process configuration: a `PortfolioBacklog`, `RequirementBacklog`, or `TaskBacklog`.

Nesting of backlog items

Can you nest backlog items in addition to using portfolio backlogs?

While you can nest backlog items, we don't recommend you do. We don't support drag-and-drop linking of nested backlog items. Instead, we support mapping of backlog items to portfolio items.

For examples of how hierarchically linked items that belong to the Requirements Category appear on the backlogs and boards, see How backlogs and boards display hierarchical (nested) items.

Hosted XML process model

Article • 02/24/2023

Azure DevOps Services

> ⓘ Note
>
> **Feature availability**: The Hosted XML process model is only supported for organizations that have migrated to Azure DevOps using the data migration tool for Azure DevOps.

The Hosted XML process model provides support for customizing work tracking objects and Agile tools for a project by modifying and importing a process template. Updates made to the process template are applied to projects that were created using that process.

> ⓘ Note
>
> For more information, see the following articles:
>
> - **Configure and customize Azure Boards**
> - **Create a project using the process of your choice**
> - **Customize your work tracking experience (process models)**
> - **Manage inherited processes**

Supported customizations

You can perform the following tasks with the Hosted XML process model.

Area

Customization support

Fields

- Add or modify a field
- Add a checkbox (Boolean) field
- Add rules to a field
- Change a field label

- Add a custom control field
- Remove a field

Pick lists

- Area paths- Iteration paths
- State or Reason fields
- Person-name field (add users)
- Add a custom pick list
- Predefined field
- Resolution State or Failure Type
- Define global lists

Work item types

- Add or modify a work item type
- Change the workflow (States, Reasons, Transitions)
- Customize the form
- Specify the WIT color
- Specify the WIT icon
- Specify the workflow state color

Backlogs and process configuration

- Add WITs to backlogs or boards
- Add portfolio backlogs
- Configure the quick add panel
- Configure the default backlog columns
- Set maximum number of Taskboard items
- Set default weekend days (Scrum)
- Set default bug management behavior
- Set default hidden backlogs
- Process configuration
- Categories

Process template

- Import a process template
- Export a process template
- Customize a process template

Add release deployment support to a work item type

The release deployments control shows release information for only those work items that have been associated to a commit that is part of a build being released. To add the release deployments control to a work item type, you need to add the following syntax to a work item type XML definition. To learn more about this control, see Link work items to deployments.

XML

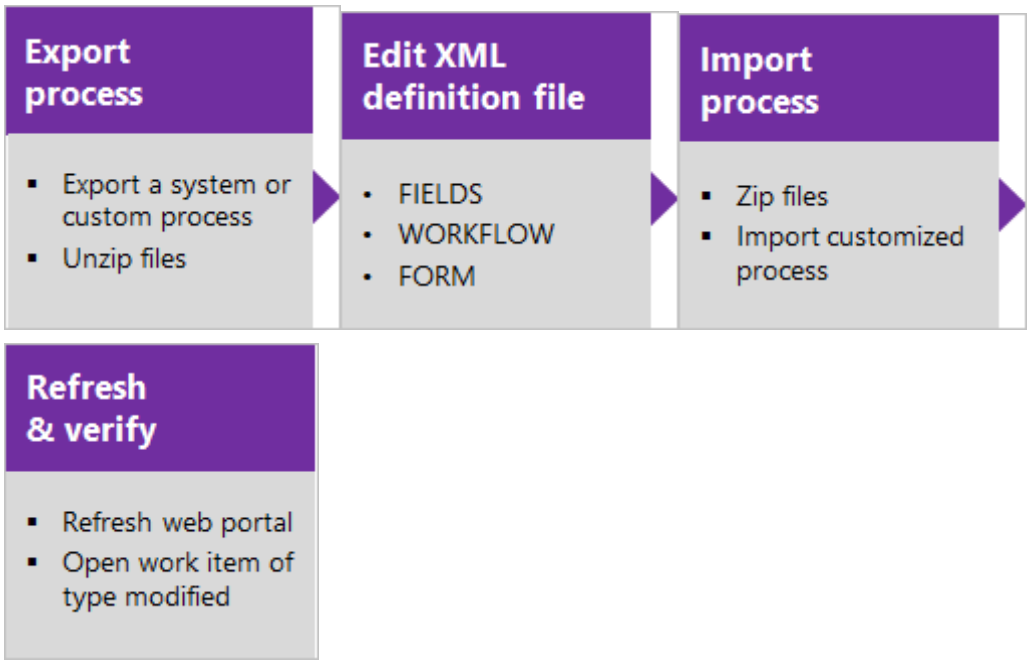
```
<Group Label="Deployment">
    <Control Type="DeploymentsControl" Name="Deployments" />
</Group>
```

This syntax has been been added to the following work item type definitions of the default process templates.

Agile	Basic	Scrum	CMMI
- Bug - Epic - Feature - Issue - Task - Test Case - User Story	- Issue - Task - Test Case	- Bug - Epic - Feature - Impediment - Product Backlog Item - Task - Test Case	- Bug - Change Request - Epic - Feature - Issue - Requirement - Review - Risk - Task

Customization sequence

Use the following sequences when you manage customizations using the Hosted XML process model. This sequence requires you to update your project by updating the process template that it uses. We recommend that you maintain your process templates in a repository for version control.



Related articles

- Customize a Hosted XML process
- Import a process
- Change a project from Hosted XML to Inheritance
- Differences between Inheritance and Hosted XML
- Work tracking object limits
- REST APIS: Processes

Process template customizations differences between Azure DevOps Services and TFS

Article • 05/12/2023

Azure DevOps Services

> **Important**
>
> Import process supports the Hosted XML process model which allows you to manage customizations through updating select XML definition files of a process template. This feature is only available for organizations that have been migrated to Azure DevOps Services using the **TFS Data Import Service**.
>
> If you use the Inheritance process model, you can customize your work tracking through the user interface by **creating an Inheritance process**. If you use the On-premises XML process model, you can customize a process template, see **Upload or download a process template** and **Customize a process template**.
>
> To learn more about process models, see **Customize work tracking**.

Azure DevOps Services uses a different model than Team Foundation Server (TFS) for relating projects and process.

- In TFS, process templates are used as starting points for projects and once a project is created, the project is the scope you customize.
- In Azure DevOps Services, process is shared across multiple projects and is the scope you customize.

The structure and overall syntax used in defining process templates remains the same, with only a few minor differences existing between templates you customize for import into Azure DevOps Services and those you upload to support an on-premises TFS.

> **Note**
>
> Migration of projects or collections from Hosted XML to the inherited model is not support for Azure DevOps Server. It is only available on the Azure DevOps Sevice.

Unsupported customizations and unreferenced plug-in files

Any reference to the following objects in any of the XML definition files result in a validation error upon import.

- Custom controls on work item forms
- Custom link types
- Global workflow
- Team field support
- For and Not rules support
- Match rule support

The following plug-ins and their associated files aren't used in defining a process, nor used to update existing projects. However, they are used to create objects or artifacts when you create a new project.

- Classification
- Work item queries (defined using the WIQL syntax)
- Test Management
- Work items

> **Note**
>
> The WIQL length must not exceed 32K characters. The system won't allow you to create or run queries that exceed that length.

The following plug-ins and their associated files are replaced by system defaults.

- Build
- Groups and Permissions
- Lab
- Version Control

The following plug-ins and their associated files are ignored.

- Microsoft Project Mappings
- Reports
- Windows SharePoint Services

Custom plug-ins aren't supported.

Object limits

When customizing a process template for import, limit the number of the objects you define as specified in Work tracking object limits.

Related articles

- Work tracking object limits
- Customize work tracking
- Differences between Inheritance and Hosted XML

Differences between Inheritance and Hosted XML

Article • 03/07/2024

Azure DevOps Services

If you are currently on Hosted XML and looking to move your process to Inherited, here is a comparison chart of the differences you can expect between the two customization models.

Expand table

Item	Inherited	Hosted XML
Process administration	Web ui and REST API	XML and import through web ui
Process security	Supported	Collection admin only
Custom work item types	Supported	Supported
Custom fields	Supported	Supported
Custom lists	Supported	Supported
Global lists	Shared picklist	Supported
Cascading picklists	With extension	Supported
Custom values in system fields	Supported	Supported
Custom reason field values & rules	Supported with work around	Supported
Custom states	Supported	Supported
Custom rules	Supported	Supported
for / not rules	Not supported	Not supported
Disable system rules	Not supported	Supported (not recommended)
Customize layout	Supported	Supported
Portfolio backlog levels	Supported	Supported

Item	Inherited	Hosted XML
Move projects to a different process	Supported	Not supported
Clone process	Supported	Not supported
Disable process	Supported	Supported
REST API	Supported	Not supported
Work item extensions	Supported	Supported
Steps (for other than Test Case)	Not supported	Supported

Related articles

- About process customization and inherited processes

Next steps

- Clone a Hosted process to an inheritance process
- Change a projects from Hosted XML to an inherited process

Feedback

Was this page helpful? Yes No

Provide product feedback

Customize a Hosted XML process

Article • 02/24/2023

Azure DevOps Services

Azure DevOps Services supports adding and updating processes through an administrative experience that is a web-based import process. After you add a process, you can create one or more projects from it. You can update the process at any time by importing it again. The changes made to the process template are then applied to all projects that use the process.

> **Important**
>
> With the Hosted XML process model, you customize work tracking by updating select XML definition files of a process template. This feature is available only when data is migrated to Azure DevOps Services by use of **Team Foundation Server Database Import Service**.
>
> To learn more about customization and process models, see **Customize work tracking**.

A process is a zip file that contains a set of interdependent files. These files define the building blocks of the work-item tracking system and other subsystems in Azure DevOps Services. Some building blocks update existing projects, while others apply only to new projects. See the following table for the full list of building blocks.

Used when importing/updating a process

Used when creating a new project

Replaced by system defaults

Ignored

Work Item Tracking

WITs

Categories

Process Configuration

Areas and Iterations

Test Management

Work Items

Work Item Queries

Build

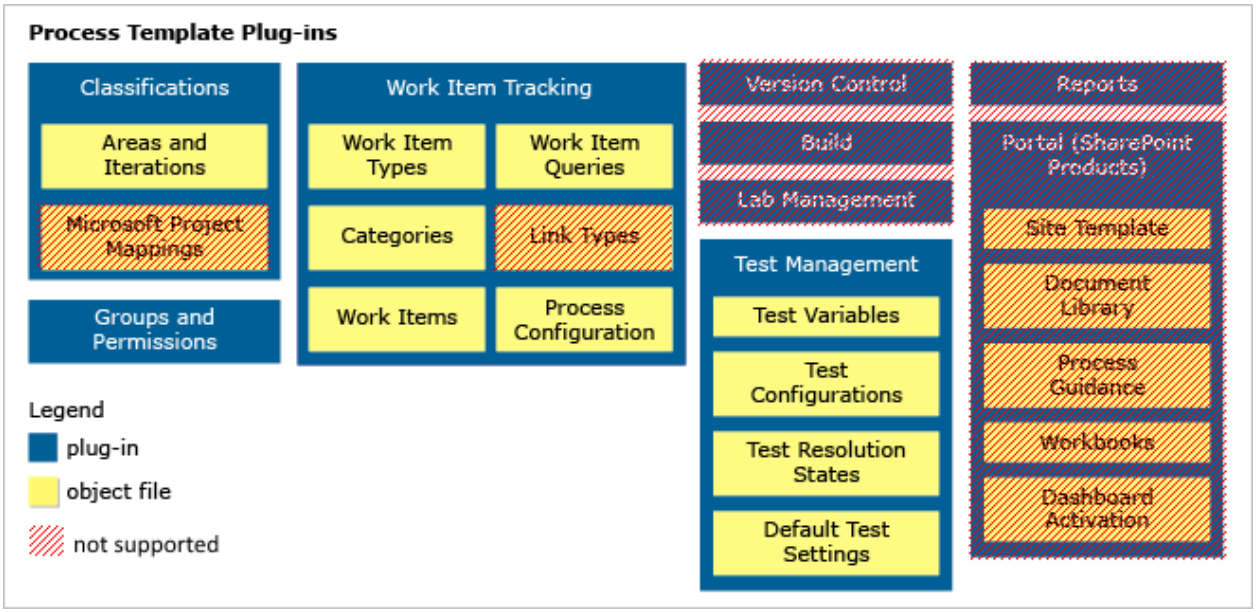
Lab Management

Version Control

Microsoft Project Mappings

Reports

Portal (SharePoint Products)



There are differences between what Azure DevOps Services supports and what on-premises Team Foundation Server supports. For a summary of these differences, see Process template customizations differences.

How to customize a process

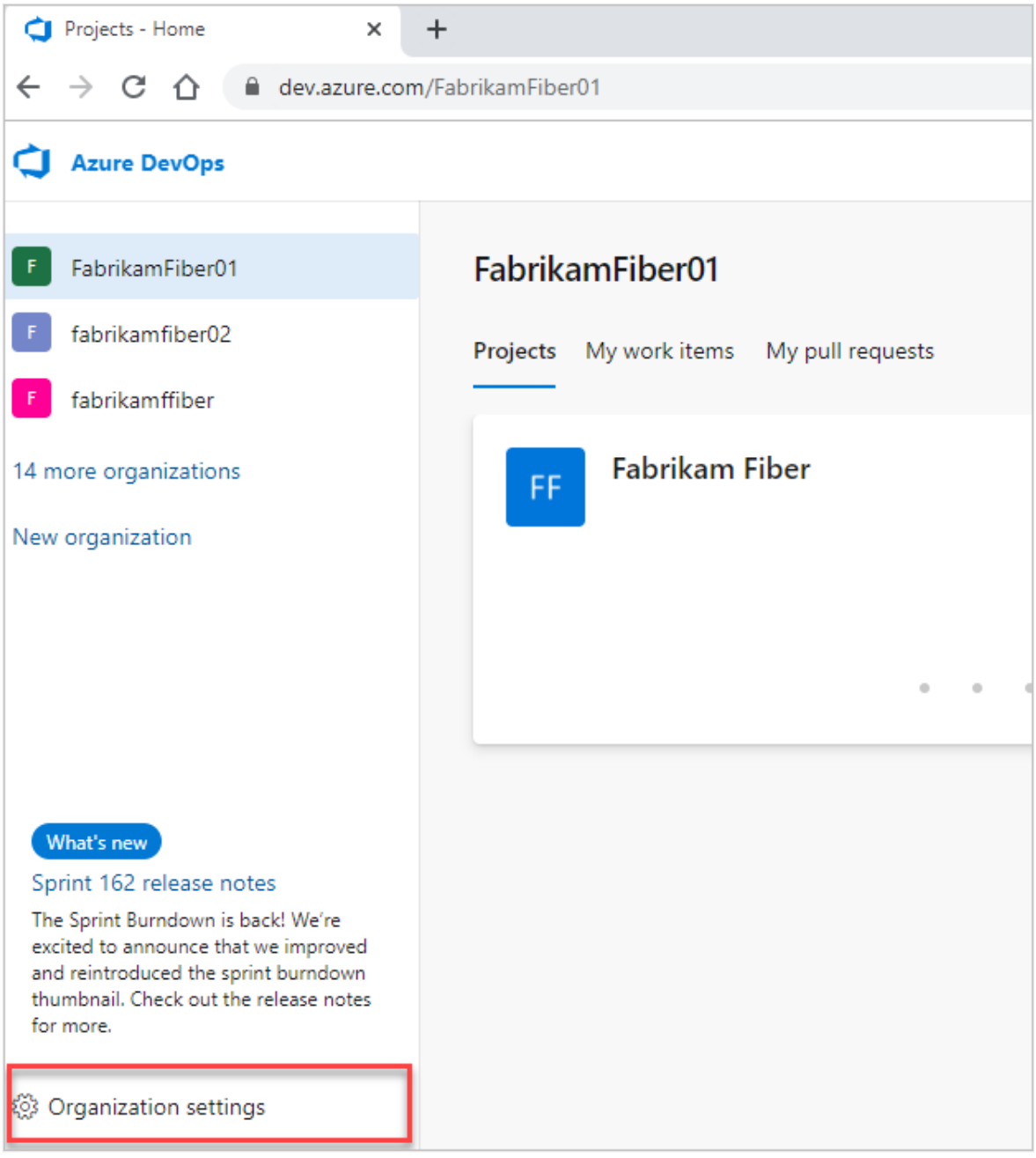
When you customize a process, starting with a well-defined process is easier than building a new one.

If you update an existing process you've used with on-premises Team Foundation Server, make sure it conforms to the constraints placed on templates for import.

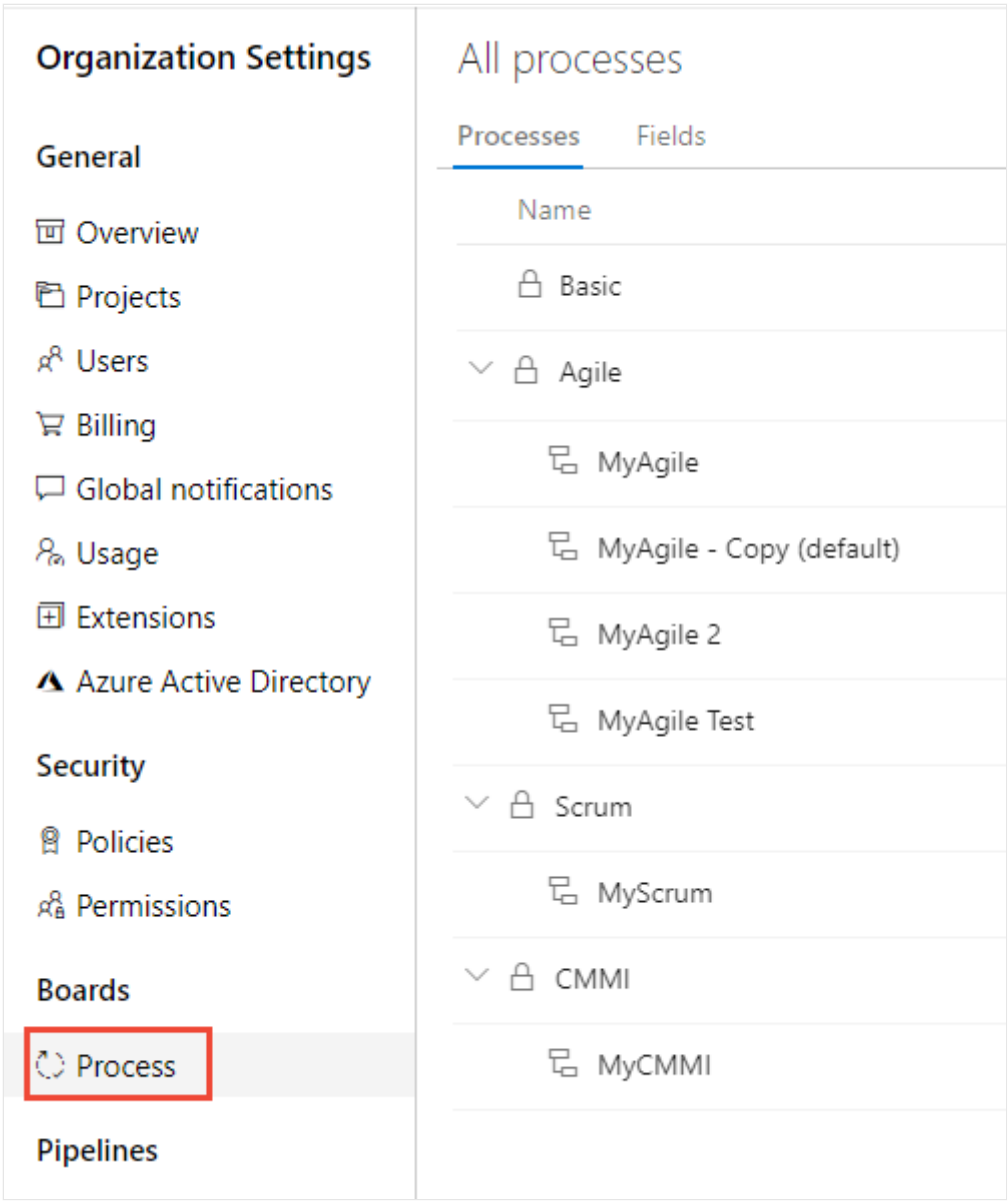
Open Settings>Process

You create, manage, and make customizations to processes from **Organization settings>Process**.

1. Choose the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.



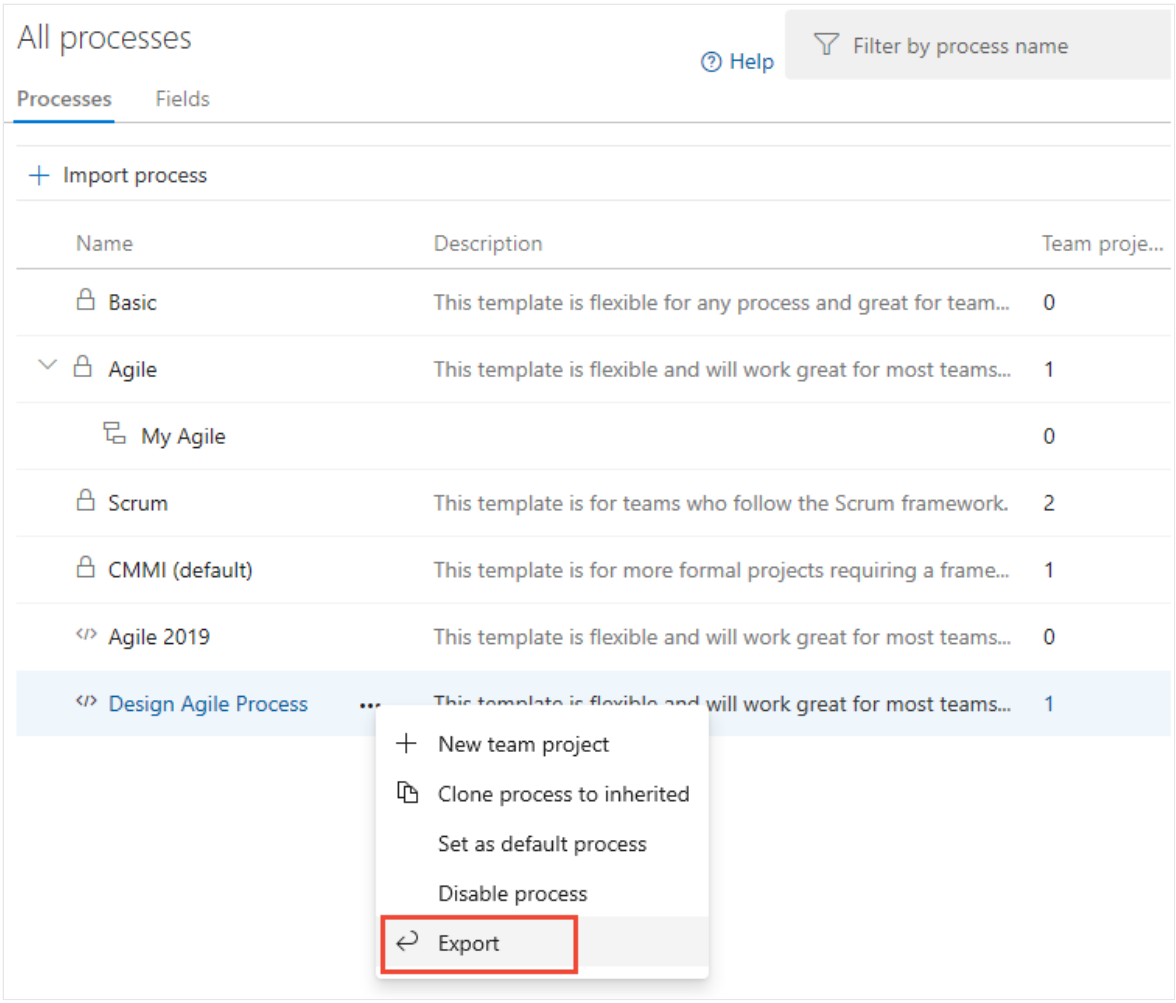
2. Then, choose **Process**.



> **Important**
>
> If you don't see **Process**, then you're working from TFS-2018 or earlier version. The **Process** page isn't supported. You must use the features supported for the On-premises XML process model.

Export and import a process

1. From the **Processes** tab, select the ellipsis (...) to open the shortcut menu for the Hosted XML process that you want to export. You can export only Hosted XML processes.



Save the zip file and extract all files from it.

2. Rename the process within the ProcessTemplate.xml file located in the root directory.

   Name the process to distinguish it from existing ones.

   ```
   <name>MyCompany Agile Process </name>
   ```

   Change the version type, and change the major and minor numbers. Provide a distinct GUID for the type as in this example:

   ```
   <version type="F50EFC58-C2FC-4C66-9814-E395D90778A3" major="1" minor="1"/>
   ```

3. Apply supported customizations.

4. Create a zip file of all files and folders in the root directory.

5. Import the zip file of your custom process.

Supported customizations

You can apply the following customizations to your process:

- Add, remove, or modify a WIT.
- Add or modify a field.
- Add up to five portfolio backlogs.
- Add categories that you'll use in your process configuration.
- Modify process configuration.
- Add global lists.

The following section lists limitations that the system imposes.

Restrictions

You can import up to 32 processes into Azure DevOps Services. Your custom processes must conform to all of the following summarized rules. Otherwise, validation error messages might appear upon import.

- Process
  - Process configuration
  - Categories
  - Work item types
    - Fields
      - Limits
      - Required fields
      - Rule restrictions
      - Consistent names and attributes
      - Identity fields
      - Workflow
      - Global lists
    - Workflow
    - Form layout

Process template

Your ProcessTemplate.xml file must conform to the syntax and rules described in ProcessTemplate XML element reference. Also, it must meet the following conditions:

- Limits the number of defined WITs to 64
- Contains only one Categories.xml definition file
- Contains only one ProcessConfiguration.xml definition file
- Uses unique friendly names across all fields and WIT definitions

Also, your process must pass the following validation checks:

- Process names are unique and contain at most 155 Unicode characters.
  - A template with the same name and version GUID as an existing process overwrites that process.
  - A template with the same name but a different version GUID generates an error.
  - Process names can't contain the following special characters: `. , ; ' ` : / \ * | ? " & % $! + = ( ) [ ] { } < >`.

    See Naming restrictions for additional constraints.
- Process folders contain no .exe files. Even if you can import a process that contains an .exe file, project creation fails.
- The process's total size is at most 2 GB. Otherwise, project creation fails.

Process configuration

The ProcessConfiguration.xml definition file must conform to the syntax and rules described in ProcessConfiguration XML element reference. Also, it must meet the following conditions:

- Specifies all **TypeFields** elements
- Is limited to five portfolio backlogs
- Contains only one unparented portfolio backlog
- Specifies only one parent portfolio backlog for each subordinate portfolio backlog
- Contains required workflow state-to-metastate mappings and doesn't reference unsupported metastates

Categories

The Categories.xml definition file must conform to the syntax and rules described in Categories XML element reference. Also, it must meet the following conditions:

- Is limited to 32 categories
- Defines all categories referenced in the ProcessConfiguration.xml file

Work item types

A **WITD** element and its child elements must conform to the syntax and rules described in WITD XML element reference. Also, it must meet the following conditions:

- There are at most 512 fields within a single WIT and 512 fields across all WITs.
- The friendly name and required **refname** attribute assigned to a WIT are unique within the set of WIT definition files.

- The required **refname** attribute value doesn't contain disallowed characters or use the disallowed namespaces System.*Name* and Microsoft.*Name*.
- Reference names contain at least one period (.), and all other characters are letters with no spaces.
- The **WITD** element contains a **FORM** element that defines a **WebLayout** element conforming to the syntax specified in WebLayout and Control elements.

Work item fields

A **FIELDS** element and its child elements must conform to the syntax and rules described in FIELD XML element reference. Also, it must meet the following conditions:

- The friendly name and required **refname** attribute assigned to a WIT are unique within the set of WIT definition files.
- The required **refname** attribute value doesn't contain disallowed characters or use the disallowed namespaces System.*Name* and Microsoft.*Name*.
- Reference names contain at least one period (.), and all other characters are letters with no spaces.

A **FIELD** element and its child elements can contain a **GLOBALLIST** element.

Limit restrictions

- A **FIELDS** element is limited to 512 fields.
- A work item type is limited to 64 person-name fields. A person-name field is one with the attribute and value `syncnamechanges=true`.
- An **ALLOWEDVALUES** or **SUGGESTEDVALUES** element is limited to 512 **LISTITEM** elements.
- A field is limited to 1,024 rules.

Required fields

The following fields are specified in the ProcessConfiguration.xml file:

- For all WITs in a category that defines a process-configuration backlog, specify the fields used for the attributes and values `type=Team` and `type=Order`.
- For all WITs in a category that defines a regular backlog or portfolio backlog, specify the field used for `type=Effort`.
- For all WITs in the category that defines the **TaskBacklog** element, specify:
  - The field used for `type=RemainingWork`.
  - The field used for `type=Activity`.

- The **ALLOWEDVALUES** rule for the field used for `type=Activity`.

Rule restrictions

In addition to the standard field-rule restrictions, the following restrictions are enforced:

- Field-rule elements can't specify the **for** and **not** attributes.
- **FIELD** elements can't contain the child-rule elements **CANNOTLOSEVALUE**, **NOTSAMEAS**, **MATCH**, and **PROHIBITEDVALUES**.
- Except for the following fields, **FIELD** definitions for System.*Name* fields can't contain field rules.
  - System.Title can contain the rules **REQUIRED** and **DEFAULT**.
  - System.Description can contain the rules **REQUIRED** and **DEFAULT**.
  - System.AssignedTo can contain the rules **REQUIRED**, **DEFAULT**, **ALLOWEXISTINGVALUE**, and **VALIDUSER**.
  - System.ChangedBy can contain the rules **REQUIRED**, **DEFAULT**, **ALLOWEXISTINGVALUE**, and **VALIDUSER**.

Consistent names and attributes

Within a process or a project collection, **name**, **type**, and other attributes that a **FIELD** element defines must be the same across all WIT definitions.

Identity fields

Identity fields correspond to fields used to contain account, user, or group names. The following core system fields are hard-coded as identity fields:

- Assigned To (System.AssignedTo)
- Authorized As (System.AuthorizedAs)
- Changed By (System.ChangedBy)
- Created By (System.CreatedBy)
- Activated By (Microsoft.VSTS.Common.ActivatedBy)
- Closed By (Microsoft.VSTS.Common.ClosedBy)
- Resolved By (Microsoft.VSTS.Common.ResolvedBy)

Add a custom identity field

A string field is recognized as an identity field when you specify the attribute **syncnamechanges** as True.

Rule restrictions on identity fields

For the current release of process import, don't specify any of the following rules within a **FIELD** definition.

- **SUGGESTEDVALUES**
- Rules that contain nonidentity values.

Correct example

To limit the account names that are valid within an identity field, specify the `VALIDUSER` element with a group name attribute.

XML

```
    <FIELD name="Project Manager" refname="Fabrikam.ProgramManager"
type="String" reportable="dimension" syncnamechanges="true">
        <ALLOWEXISTINGVALUE />
        <VALIDUSER group="[PROJECT]\Program Manager Group" />
        <HELPTEXT>The program manager responsible for signing off on the
user story.</HELPTEXT>
    </FIELD>
```

Before you import the process, make sure you've created the group in the projects that the process updates.

Incorrect example

The following example isn't valid because it specifies:

- An `ALLOWEDVALUES` element.
- A `DEFAULT` element that specifies the nonidentity string `value="Not Assigned"`.

XML

```
    <FIELD name="Project Manager" refname="Fabrikam.ProgramManager"
type="String" reportable="dimension" syncnamechanges="true">
        <ALLOWEXISTINGVALUE />
        <ALLOWEDVALUES>
          <LISTITEM value="[PROJECT]\Program Manager Group" />
          <LISTITEM value="Not Assigned" />
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="Not Assigned" />
        <VALIDUSER />
        <HELPTEXT>The program manager responsible for signing off on the
user story.</HELPTEXT>
    </FIELD>
```

Workflow

A **WORKFLOW** element and its child elements must conform to the syntax and rules described in WORKFLOW XML element reference. Also, it must meet the following conditions:

- Limits each WIT to 16 workflow states
- Defines all workflow states that are mapped to metastates in the ProcessConfiguration definition file
- Defines a transition between all workflow states mapped to the "Proposed" state category and workflow states mapped to the "InProgress" state category
- Defines a transition between all workflow states mapped to the "InProgress" state category and workflow states mapped to the "Complete" state category.

For a description of state category and mappings, see Workflow states and state categories.

Global lists

For the Hosted XML process model, the following limits are placed on global-list import:

- There are at most 64 global lists.
- There are at most 512 items per list.
- Approximately 10,000 items can be defined in total among all global lists that are specified across all WITs.

Form layout

A **FORM** element and its child elements must conform to the syntax and rules described in FORM XML element reference.

A **Control** element can't specify a custom control. Custom controls aren't supported.

Related articles

- Import and export a Hosted XML process
- Rules and rule evaluation
- Change a project from Hosted XML to an inherited process
- Clone a Hosted XML process to an Inheritance process
- Supported operations when moving from Hosted XML to an inherited process

Import and export a Hosted XML process

Article • 02/24/2023

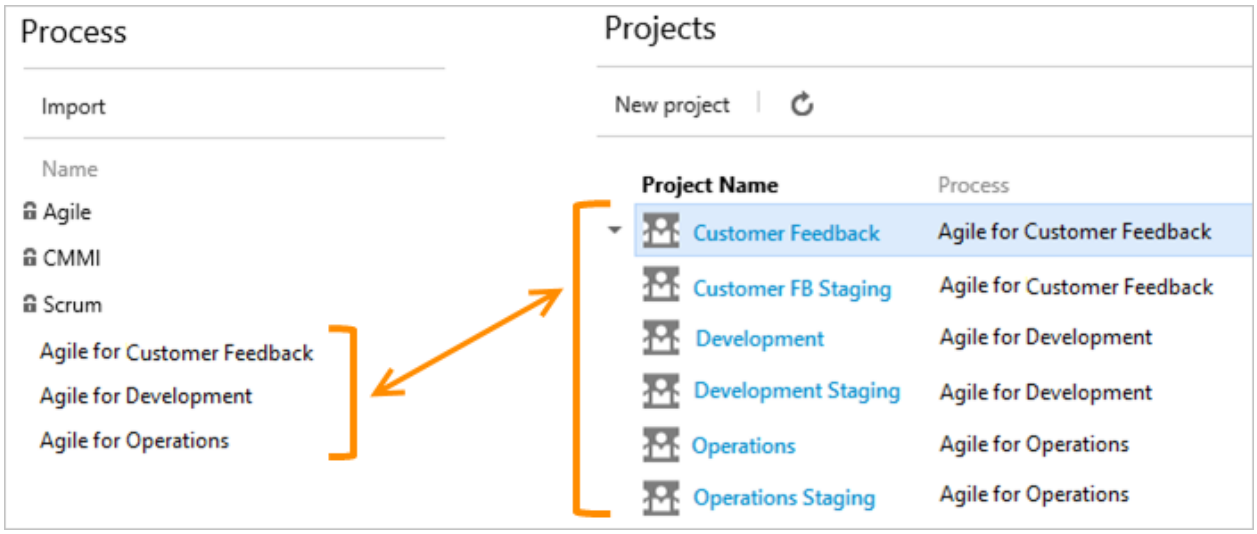
Azure DevOps Services

> **Important**
>
> Import process supports the Hosted XML process model which allows you to manage customizations through updating the WIT definition of a process template. This feature is only available for organizations that have been migrated to Azure DevOps Services using the **TFS Database Import Service**.
>
> If you use the Inheritance process model, you can customize your work tracking through the user interface by **creating an inherited process**. If you use the On-premises XML process model, you can customize a process template, see **Upload or download a process template** and **Customize a process template**.
>
> To learn more about process models, see **Customize work tracking**.

In Azure DevOps Services, you manage customization of work tracking objects through a process. Hosted XML processes are imported and exported through a web based administration experience.

When you import a new process, you're able to create new projects using it.
When you import an existing process, all projects that use that process are updated to reflect the changes.

For example, updates made to any of the custom processes listed below automatically update the two projects that use the process.



As you discover additional customizations that you need to make, you can export the existing process, update the process XML definition files, zip the result and re-import the newly updated process. Your changes are applied to all your existing projects that use that process.

Import process supports the following scenarios:

- Import an existing process from an on-premises Azure DevOps Server
- Import a new process created from an existing exported process
- Import an update to an existing process, and have all projects using that process be updated

Import a process from an on-premises Azure DevOps

If you want to test your process in an on-premises Azure DevOps Server (formerly named Team Foundation Server (TFS)) to see if it works in Azure DevOps Services, perform the steps that follow.

In an on-premises Azure DevOps Server, each project has its own copy of a process so careful consideration needs to be made as to which processes should exist for your organization. Migration to Azure DevOps Services is a great time to think about process alignment across your organization and reducing the number of variants of that centralized process.

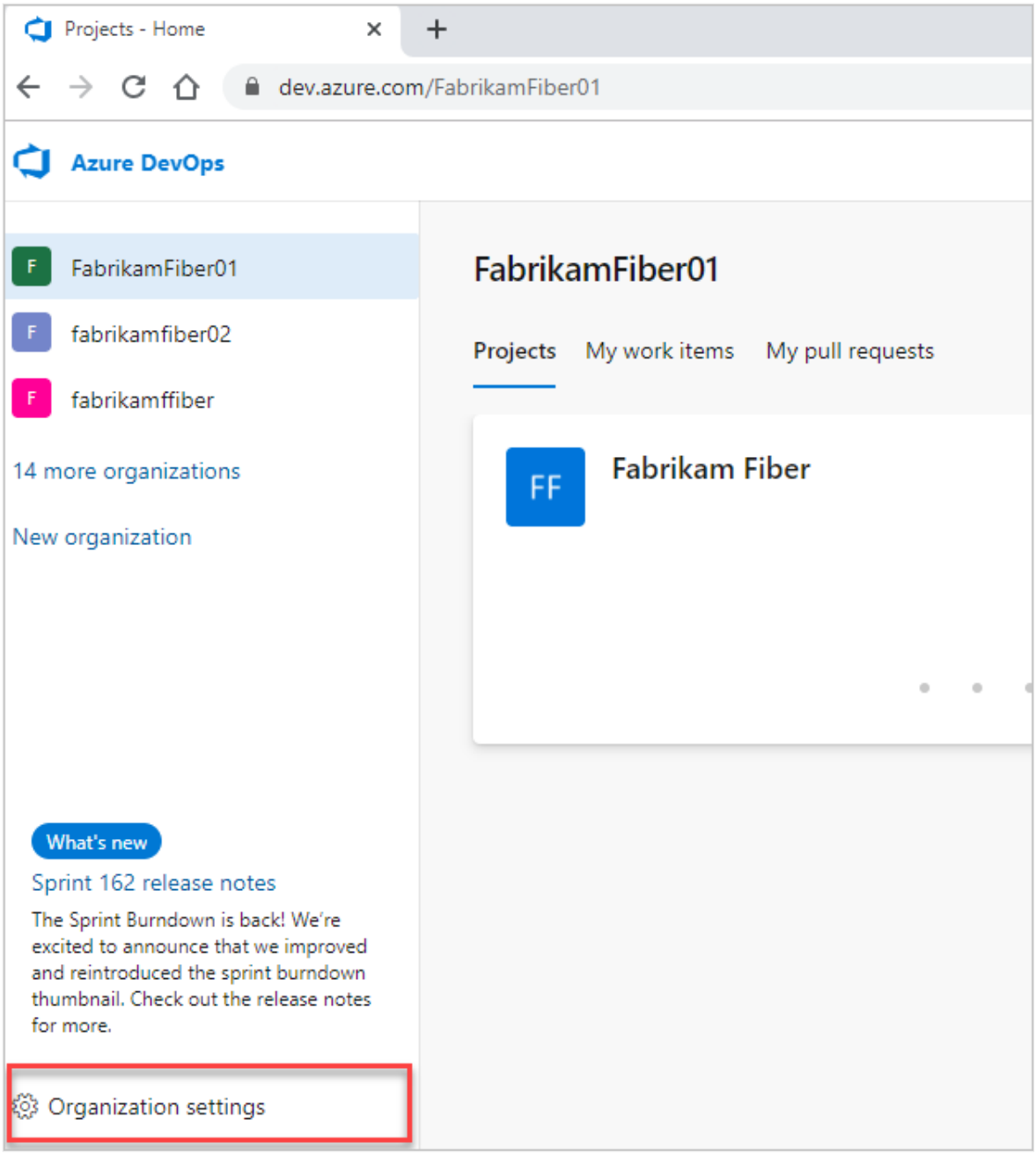
1. Run the process export script to generate a process for a given project.
2. (Optional) Edit the ProcessTemplate.xml file name and description. Ensure it conforms to the rules and constraints outlined in Customize a process.
3. Create a zip file of the process folder and files.

4. Import the zip file of your custom process by following the steps in the next section.
5. Repeat the above process for each process you want to import to Azure DevOps Services.
6. Using the imported process(es), create projects on Azure DevOps Services for each you want to migrate.

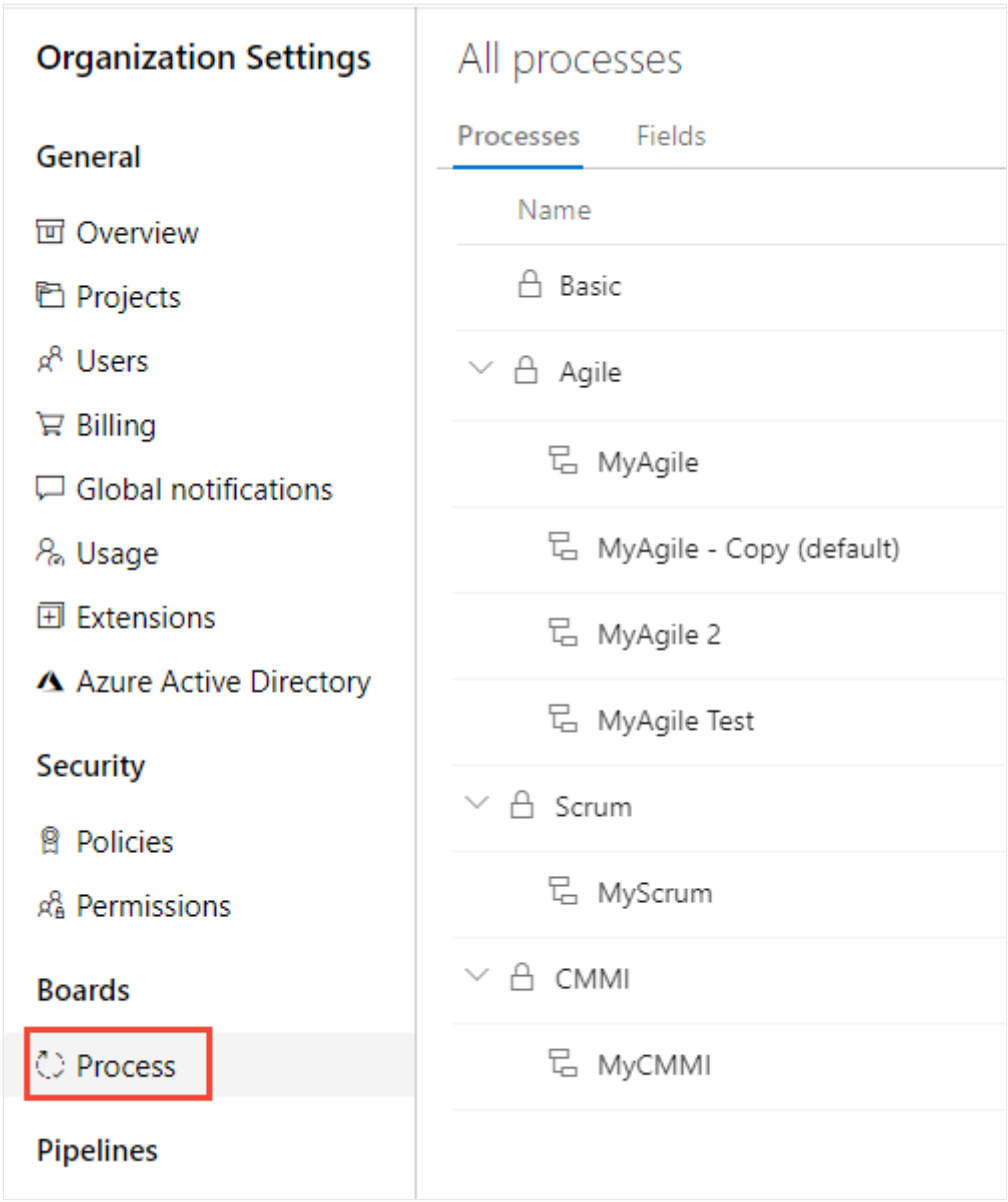
Open Settings>Process

You create, manage, and make customizations to processes from **Organization settings>Process**.

1. Choose the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.



2. Then, choose **Process**.



> **Important**
>
> If you don't see **Process**, then you're working from TFS-2018 or earlier version. The **Process** page isn't supported. You must use the features supported for the **On-premises XML process model**.

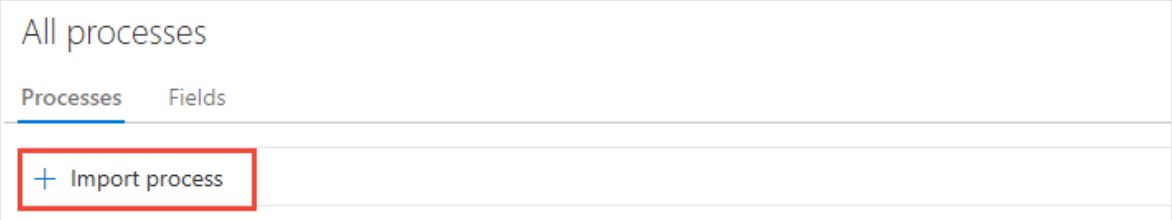
Import a process

Prior to importing a process, you'll want to customize it to support your work tracking needs. Name your process something other than Scrum, Agile, or CMMI. These system processes are locked and you're prevented from overwriting them.

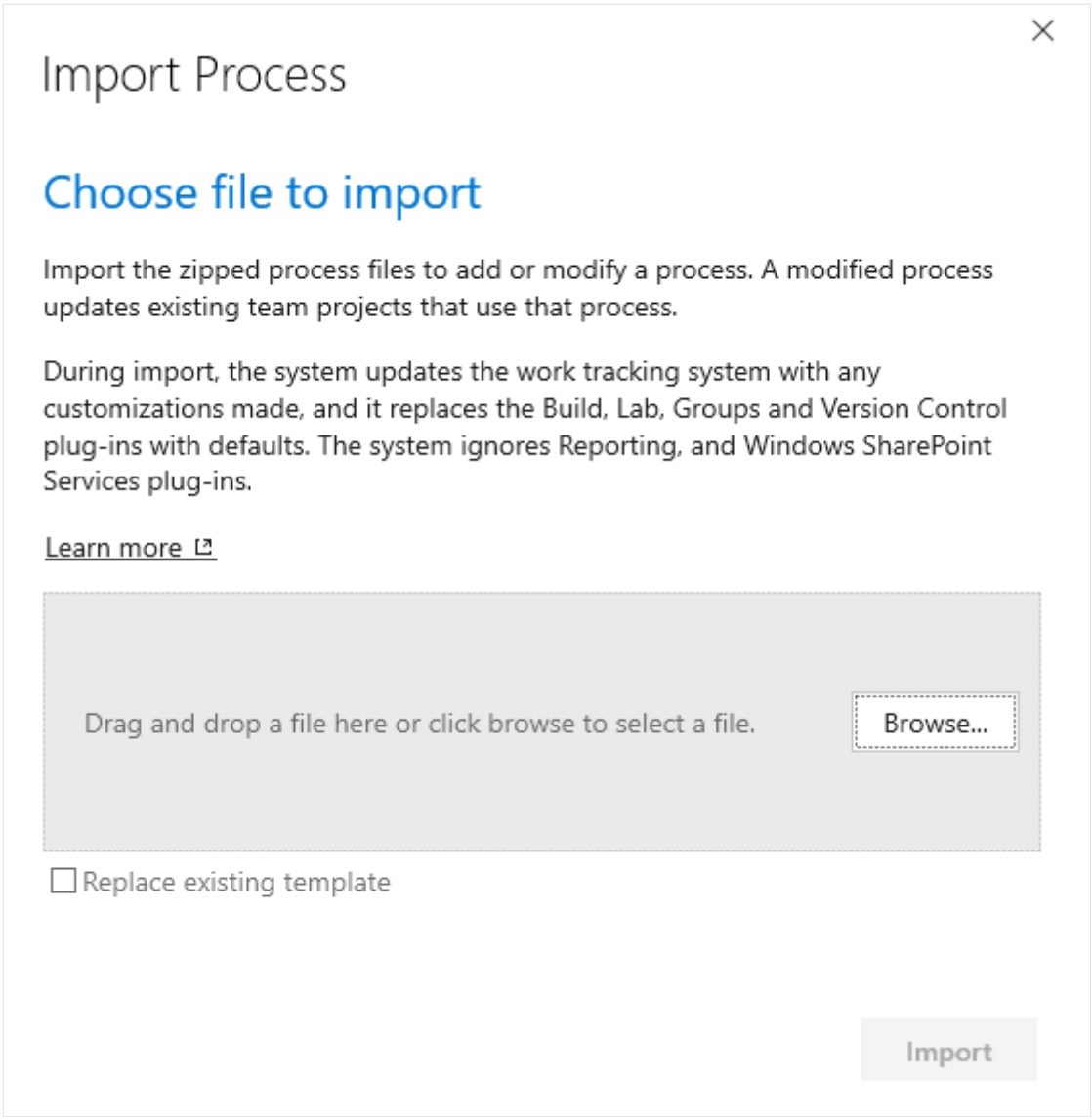
1. Only members of the **Project Collection Administrators** group can import a process. To learn more, see Change project collection-level permissions.

2. From the **Processes** tab, choose import and then drag-and-drop or browse to the zip file of the process you've customized.

> **Note**
>
> If you don't see the **Import process** link, then your organization isn't set up to support the Hosted XML process model. You should work with the **Inheritance process model** for your customization needs. The Hosted XML process model is only supported if your account was created through the **Data Import Service**.



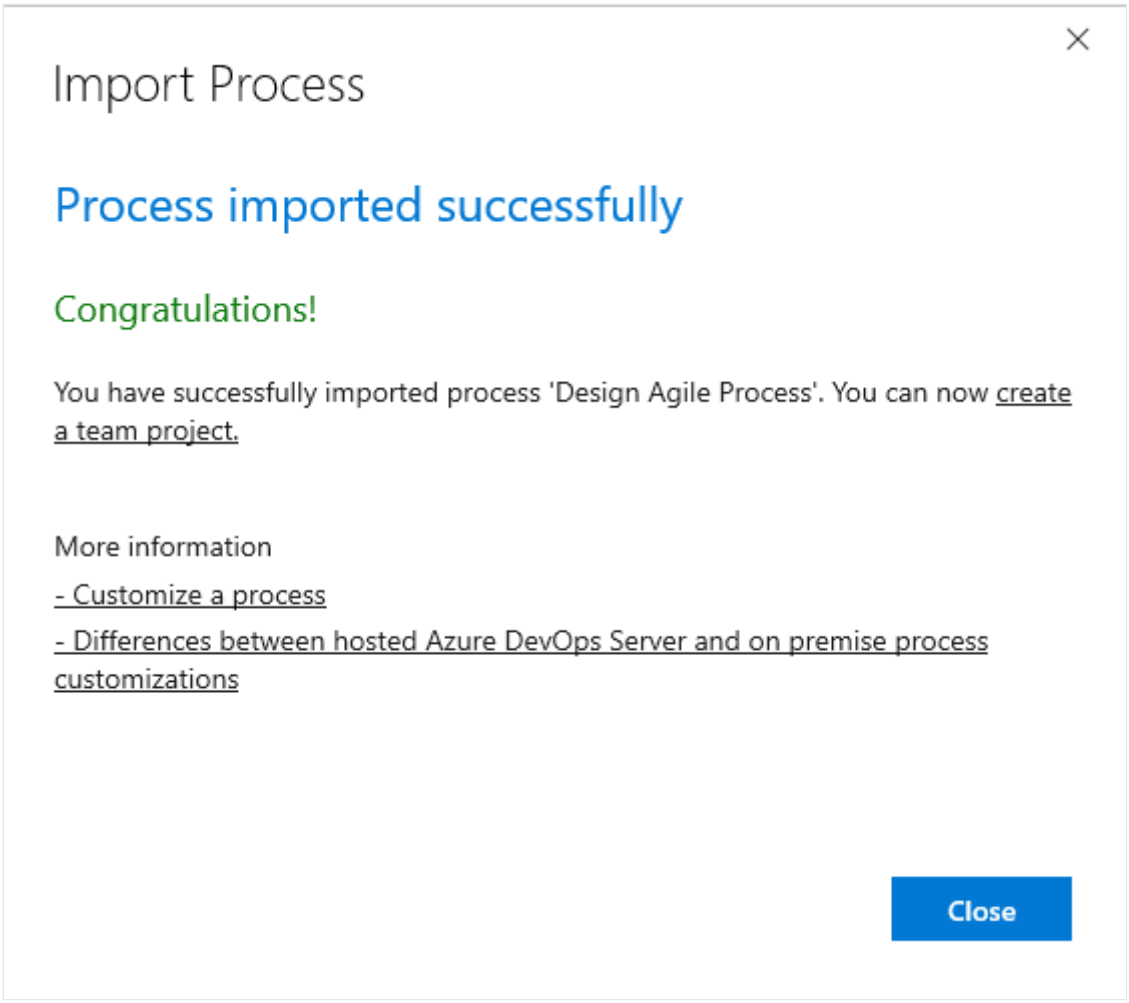
3. Choose the file to upload. Your custom process must meet specific constraints in order to pass validation checks during import.



Check the **Replace existing template** if you are updating an existing template. The import process overwrites any template that has the same name as the one you import and requires that you check this box to confirm you want it replaced.

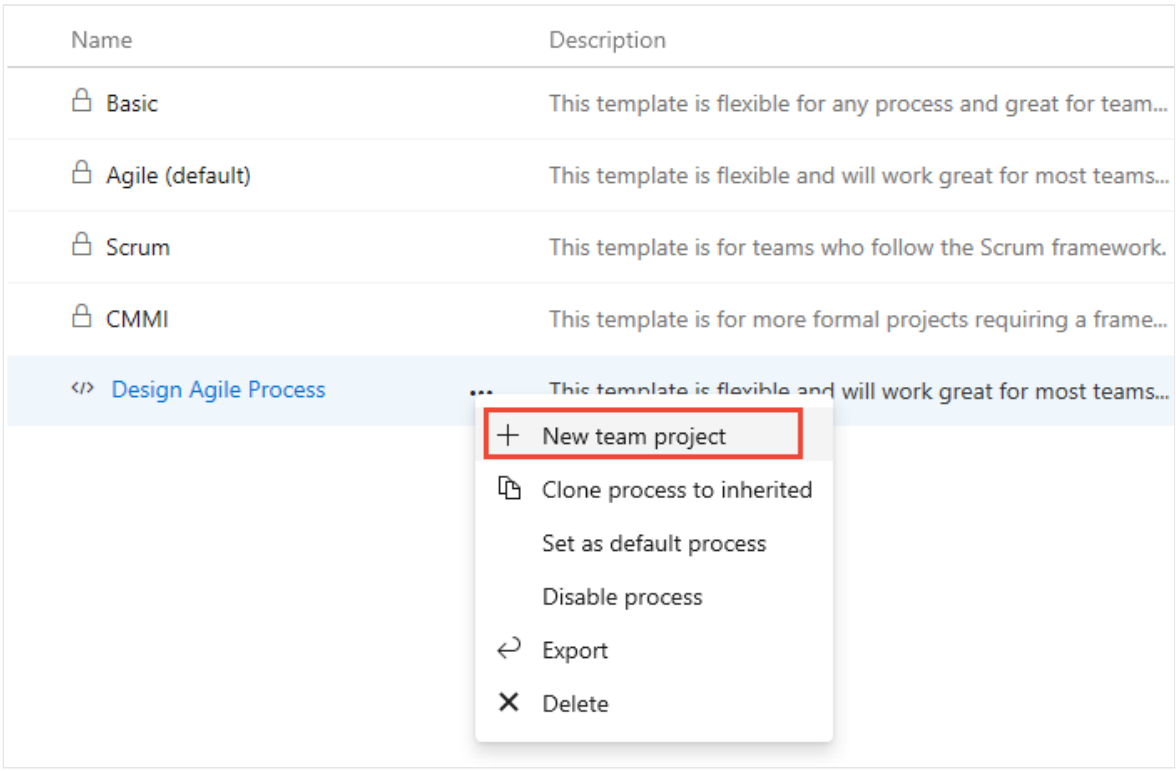
> **Important**
>
> You can't update one of the locked processes: Agile, CMMI, and Scrum.

4. Upon successful import, you'll see the following message.

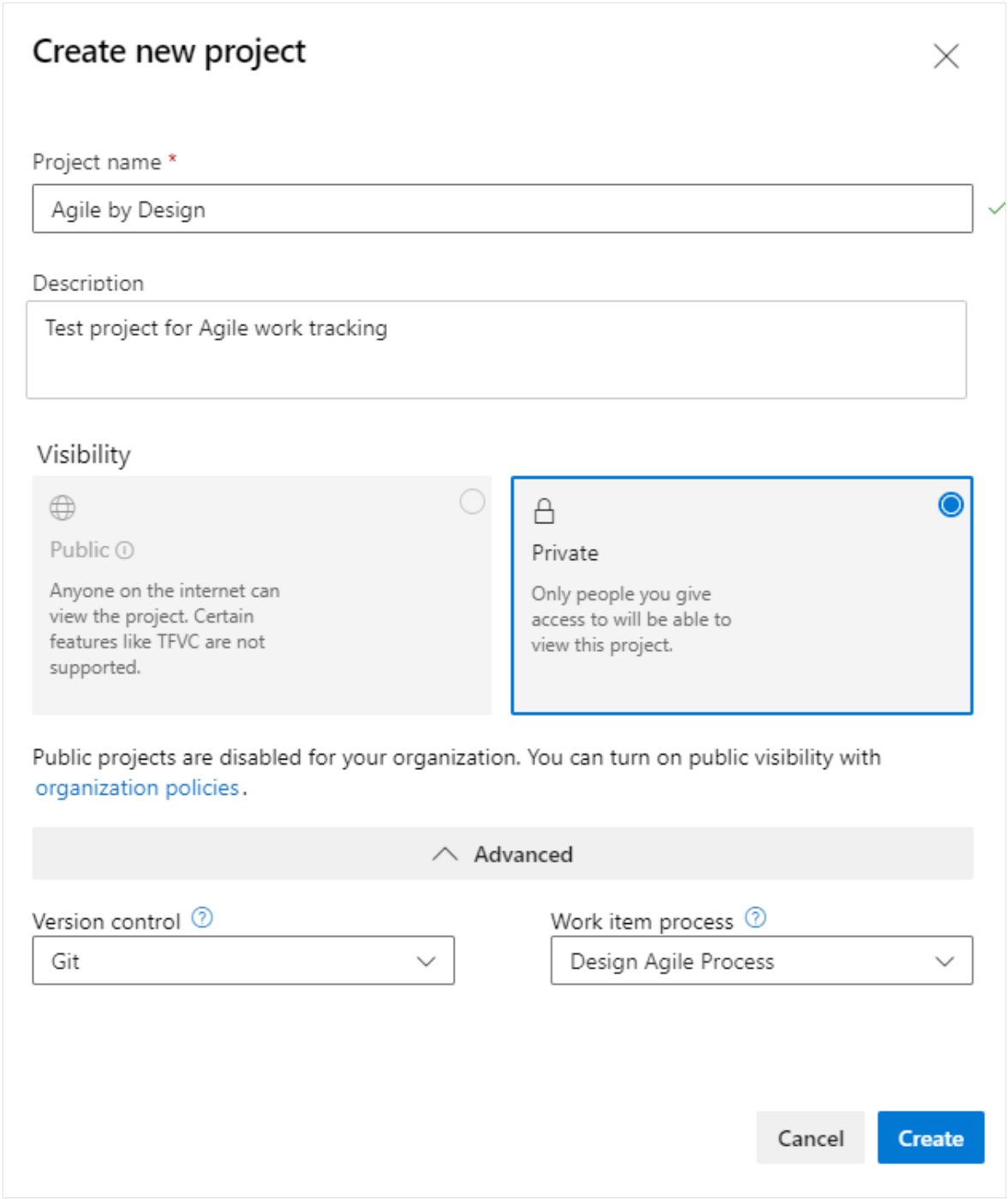


If the process doesn't pass the validation checks performed on import, you'll receive a list of error messages.
Correct each error and then retry the import.

5. You can immediately create a project using the newly imported process.



6. Fill out the form that appears. To learn more about the different options, see Create a project.

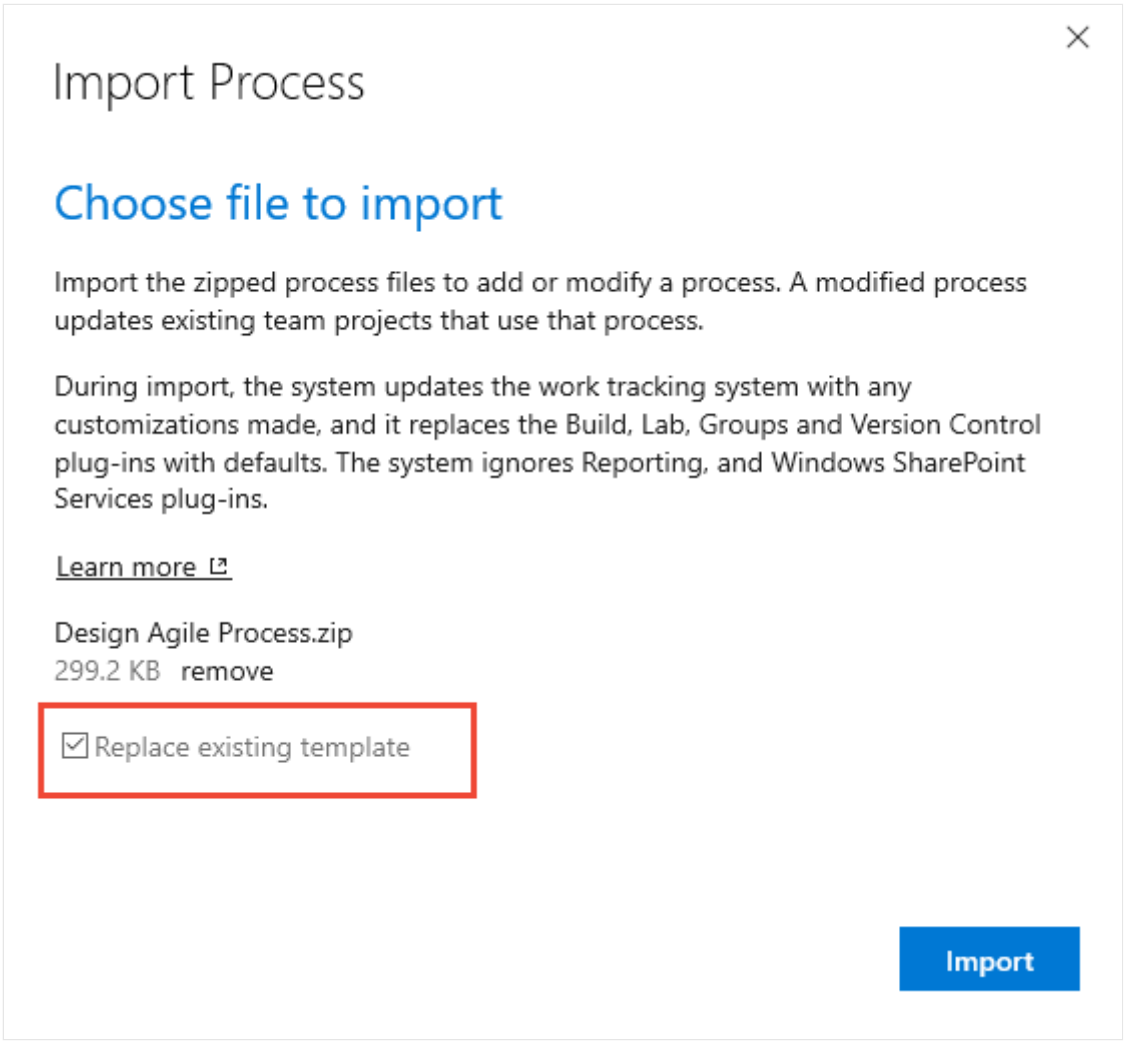


Update an existing process

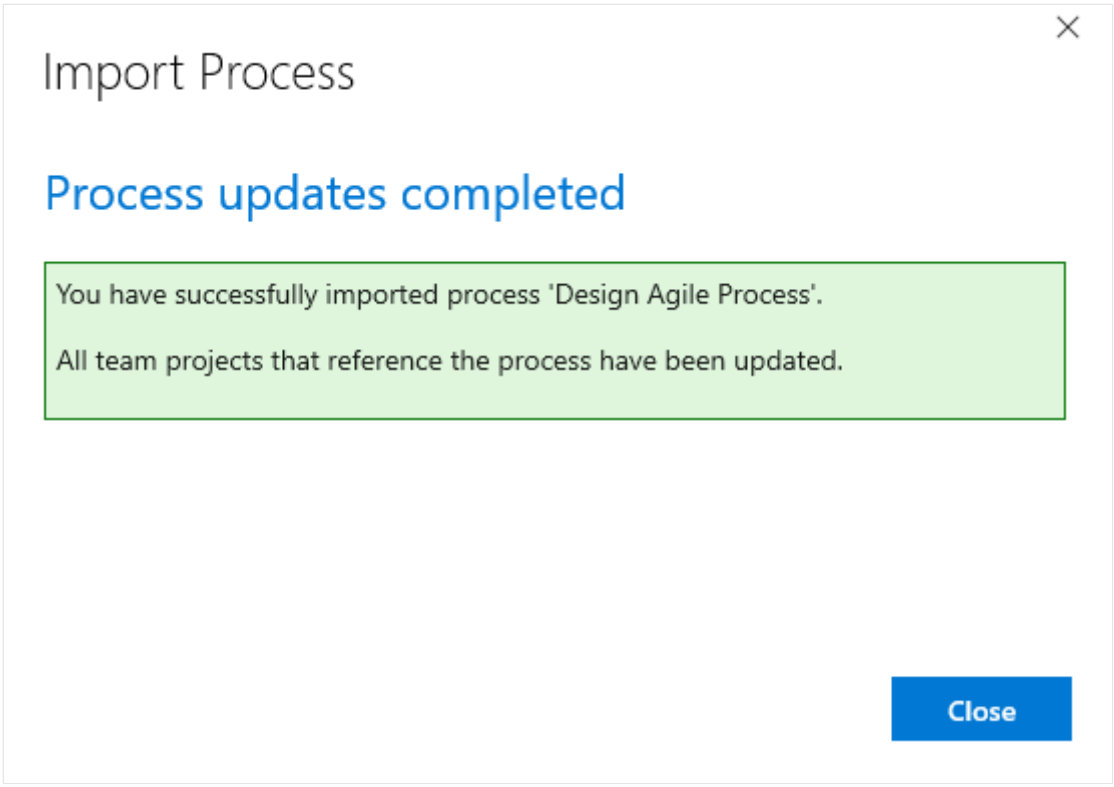
Once you've added a process, you can update it by importing a zip file where you've modified one or more files within the process template.

> **Note**
>
> It's a best practice to **Export a process** before making changes so that you don't accidentally overwrite changes made by other users.

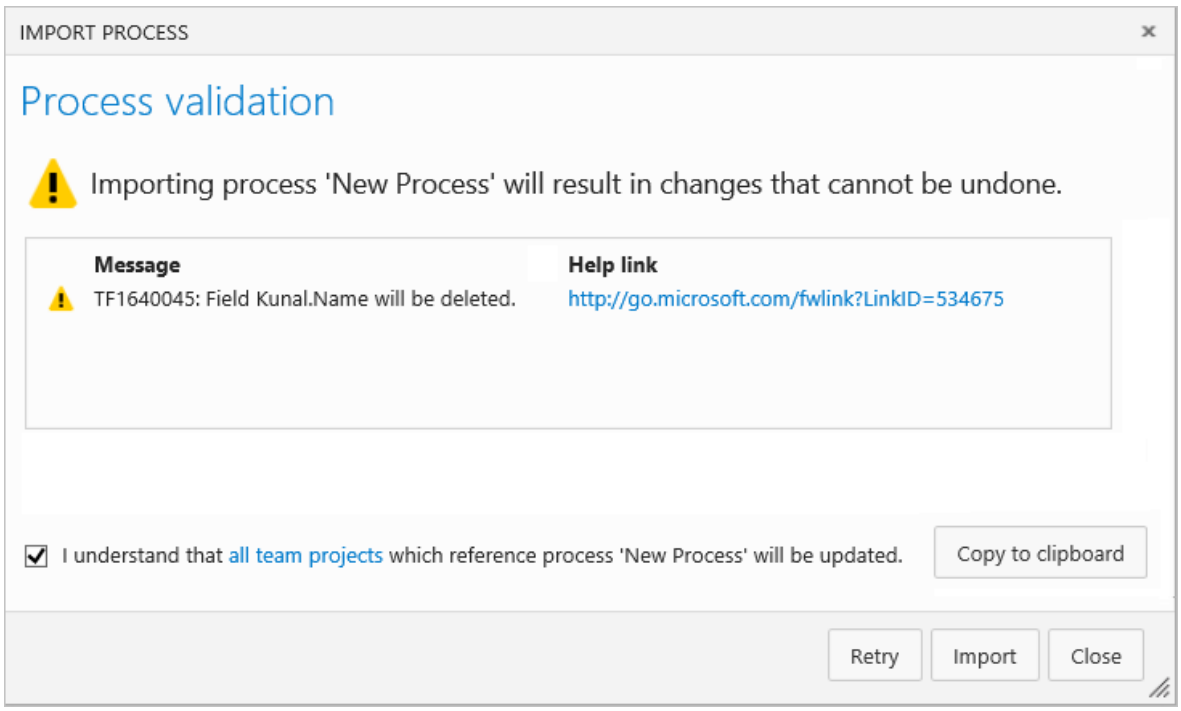
1. Import the process according to steps 2 and 3 from the previous procedure.
2. Check the **Replace existing template** to indicate you want to overwrite the existing process.



3. The Import Process dialog indicates that the system is updating projects that reference the Hosted XML process.
4. Upon successful import, you'll see the following message. All projects that were created with the process are updated with the modifications.

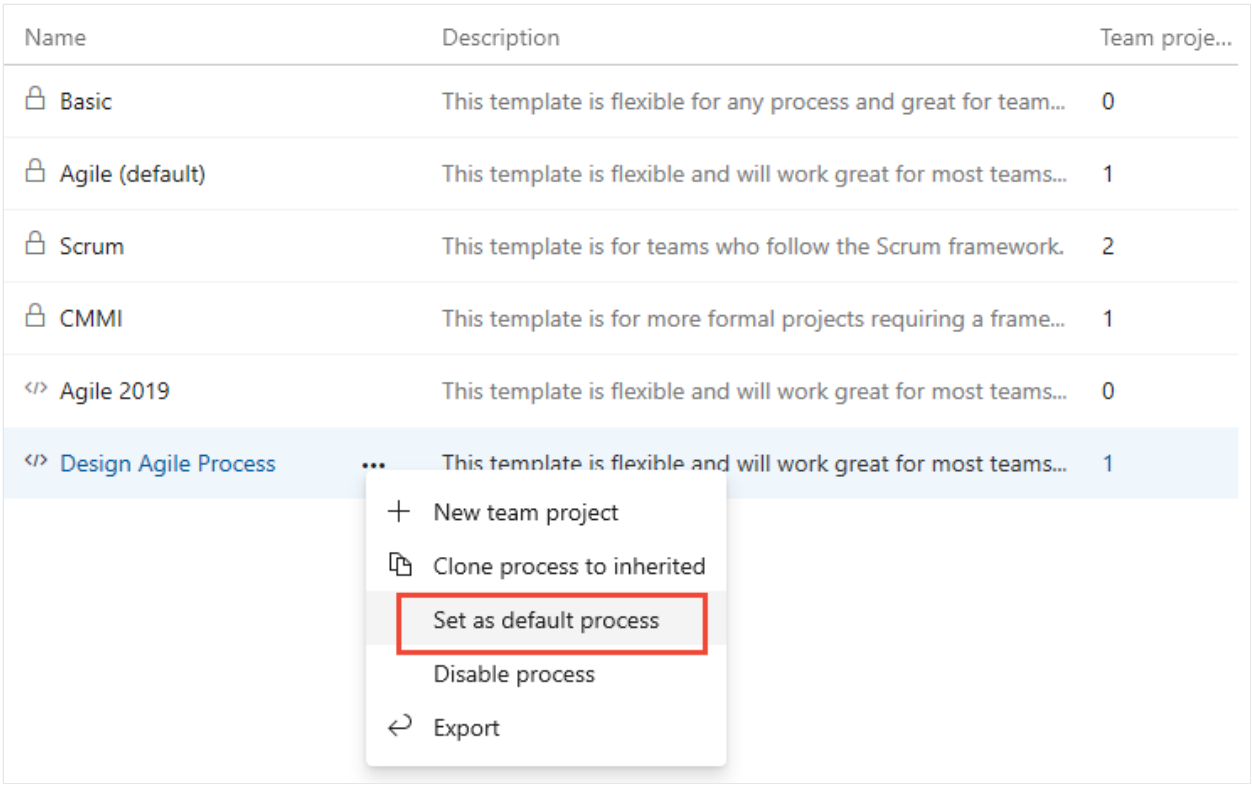


5. If you renamed or deleted fields or work item types, you receive a confirmation message. Go ahead and check the box and proceed with the import. To learn more about each message, select the forward link provided. Information messages don't require any action on your part.



Set the default process

Set a process as the default to have it preselected for all new projects you plan to create.



Export a process

Export a process when you want to update it or use it as the basis for creating another process.

The system exports a zip file containing an XML representation of the process. This can be modified and later imported.

Related articles

- Customize a Hosted XML process
- Change a project from Hosted XML to an inherited process
- Clone a Hosted XML process to an Inheritance process
- Supported operations when moving from Hosted XML to an inherited process

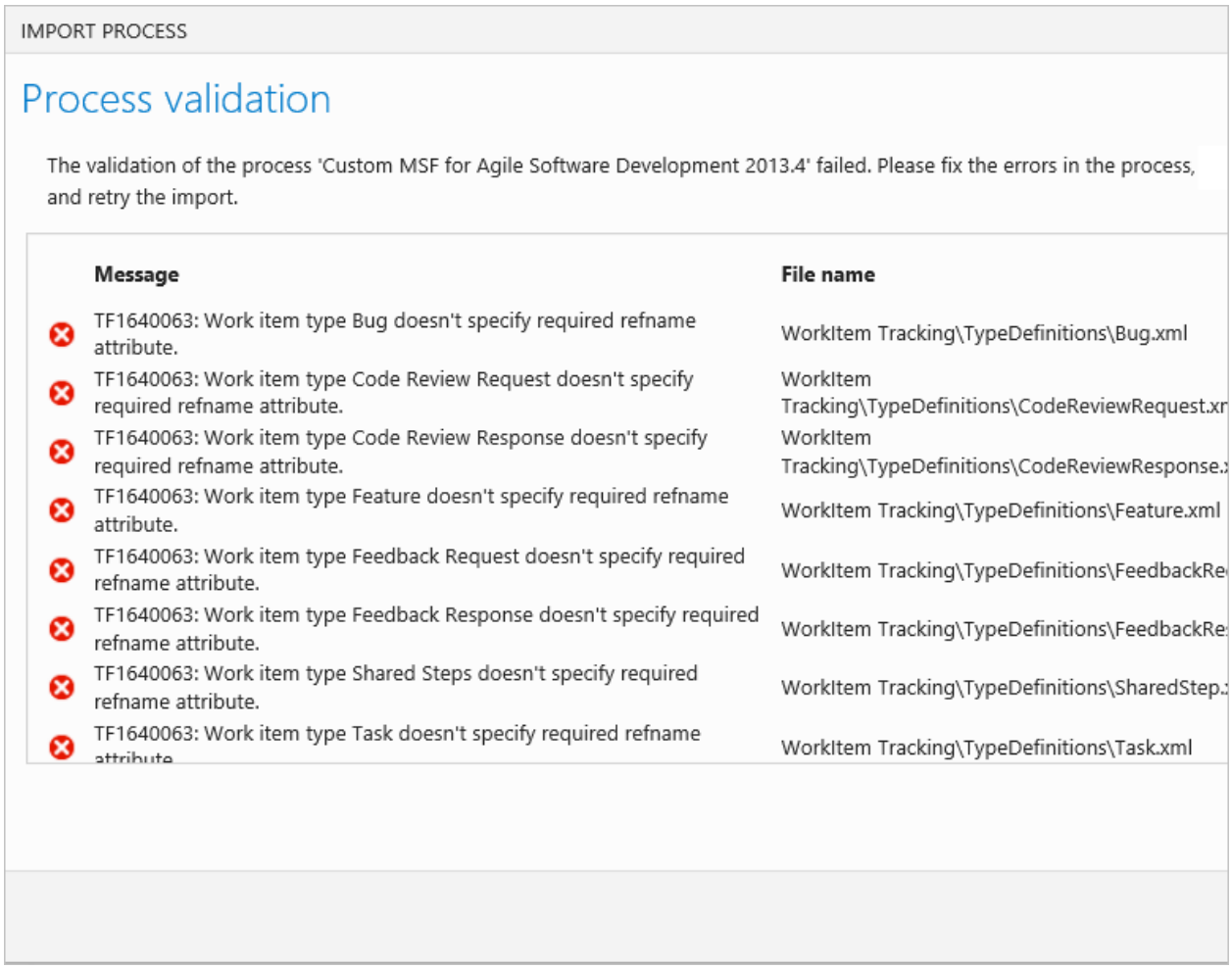
Resolve validation errors for process import

Article • 02/24/2023

Azure DevOps Services

> **Important**
>
> Import process supports the Hosted XML process model which allows you to manage customizations through updating the WIT definition of a process template. This feature is only available for organizations that have been migrated to Azure DevOps Services using the **Azure DevOps data import service**. **Contact us** if you have any questions about Azure DevOps Services process customization.
>
> If you use the Inheritance process model, you can customize your work tracking through the user interface by **creating an inherited process**. If you use the On-premises XML process model, you can customize a process template, see **Upload or download a process template** and **Customize a process template**.
>
> To learn more about process models, see **Customize work tracking**.

During process import, the process is validated to ensure the system works as expected for the custom process. You'll receive a list of error messages if the process fails validation.



If you've received a validation error when you tried import process, you'll need to resolve the error before retrying the import. Each error has a link to learn more about the specific validation failure and guidance on how to correct it. Apply the resolution fix(es) provided for the message(s) that you received, zip up the updated files, and retry the import operation.

If you're just starting your customization, review the validation rules provided.

Information messages - no action required

TF402555: Field *[refName]* is deleted.

The process that you're importing doesn't include the named field in the process being updated, therefore it is removed as part of the update operation.

Scenario example

1. Add custom field to Bug.xml work item

```
XML
```

```
<FIELD name="Foo" refname="MyCompany.CustomFields.Foo" type="String"
reportable="dimension" />
```

2. Import process
3. Remove field from Bug.xml
4. Import updated process

The system displays an information message about the field to be deleted.

`TF402555: Field MyCompany.CustomFields.Foo will be deleted`

TF402591: Field with reference name *[refName]* cannot be renamed to '*[name1]*' from '*[name2]*' in existing processes *[processName]*.

The process that you're importing contains a renamed field that uses the same `refname` in the current process(es).

Field names cannot be renamed.

TF402598: Work item type *[refName]* is deleted.

The process that you're updating doesn't include the named WIT that exists in the current process. As part of updating the existing process, the system deletes the named WIT from the current process.

Scenario example

1. Create new work item type called "LSI"

```
XML

<WORKITEMTYPE name="LSI" refname="My.LSI">
```

2. Import process
3. Remove the LSI work item type from the process
4. Import updated process

The system displays an information message about the deletion.

`TF402598: Work item type My.LSI will be deleted`

TF402601: Work item type *[witName]* is renamed to *[name1]* from *[name2]*.

The process contains a renamed WIT. The WIT is renamed in the existing process.
The process that you're updating contains a WIT that's been renamed from the name in the current process.
As part of updating the existing process, the system renames the WIT in the current process. All work items in existing projects that reference the process are also renamed.

Scenario example

1. Create new work item type called "LSI"

   ```
   XML

   <WORKITEMTYPE name="LSI" refname="My.LSI">
   ```

2. Import process
3. Rename the LSI work item type to Live Site Incident

   ```
   XML

   <WORKITEMTYPE name="Live Site Incident" refname="My.LSI">
   ```

4. Import updated process

The system displays an information message about the WIT to be renamed.
`TF402601: Work item type My.LSI will be renamed to 'Live Site Incident' from 'LSI'`

Miscellaneous file errors

TF402586: Error parsing file: *[Error message]*

Malformed XML syntax can cause parsing file errors. Missing closing tags, missing quotes, missing open or close brackets (< or >) can cause a parsing file error.

From the error message, determine and correct the source of the malformed XML.

TF402594: File violates the schema with the following error: *[Error message]*

Each XML file in the process zip file must conform to the given schema. Schema violation errors are caused by custom XML tags or attributes within the XML file. Read the error message to determine the source of schema violation and fix accordingly.

The schema definition for work item tracking defines all child elements within the `FORM` element as camel case and all other elements as all capitalized. If you encounter errors when validating your type definition files, check the case structure of your elements. Also, the case structure of opening and closing tags must match according to the rules for XML syntax.

Error examples

Custom XML tag:

```XML
<WORKITEMTYPE name="Bug" refname="My.Bug">
  <FOO>Hello World</FOO>
  ...
```

Extra attribute added to XML element:

```XML
<WORKITEMTYPE name="Bug" refname="My.Bug" foo="hello world">
```

VS412450: Xml schema validation error in *[fileName]*. *[Error message]*

Correct the schema error in the named file.

Process template plug-in errors

The ProcessTemplate.xml is the root file that defines the entire process and all XML definition files that will be imported to add or update a process. This file contains all of the plug-ins and task groups that are referenced when creating a project. Each task group references a subordinate XML file (often in a subfolder) where the specific tasks are defined. In general, you specify one task group for each plug-in.

The ProcessTemplate.xml definition file must conform to the syntax and rules described in ProcessTemplate XML element reference.

VS412457: Specified file '*[fileName]*' could not be found in the process package.

Correct the process zip package to include the named file.

VS402452: The process template did not specify a version, or specified an invalid version.

Edit the ProcessTemplate.xml file to specify a version.

Error example

ProcessTemplate.xml file specifies the same `version` GUID as for the Agile process, which is a locked process.

XML

```
<ProcessTemplate>
  <metadata>
    <name>Fabrikam Agile</name>
    <description>Use this template to support Fabrikam Agile planning
methods.</description>
    <version type="ADCC42AB-9882-485E-A3ED-7678F01F66BC" major="7"
minor="36" />
```

Resolution example

A different GUID is specified.

XML

```
<ProcessTemplate>
  <metadata>
    <name>Fabrikam Agile</name>
    <description>Use this template to support Fabrikam Agile planning
methods.</description>
    <version type="7710F7A4-1F19-4054-9FBC-D94A5935221E" major="7" minor="1"
/>
```

VS412453: Process template zip file doesn't contain required plugin *[pluginName]*.

All files listed within the ProcessTemplate.xml file and its supported plug-in files must be present within the process zip file.

Use a search tool to find all instances of filename=value within the set of process files and folders. Then, either update the plug-in to remove the missing named file, or add the named file to the process folder where it belongs.

XML

```
 <taskList filename="WorkItem Tracking\WorkItems.xml" />
```

Error example

The WorkItemTracking plug-in specifies `fileName="WorkItem Tracking\TypeDefinitions\Epic.xml`, however it hasn't been added to the WorkItem Tracking\TypeDefinitions folder.

XML

```
     <WORKITEMTYPE fileName="WorkItem Tracking\TypeDefinitions\Epic.xml" />
```

Resolution example

Add the Epic.xml file to the WorkItem Tracking\TypeDefinitions folder.

VS412454: Plug-in *[pluginName]* contains several `taskList` {1}, {2}. Only one `taskList` per plugin is allowed.

Correct the ProcessTemplate.xml file for the named plug-in to reduce the number of `tasklist` elements defined.

Error example

The WorkItemTracking plug-in contains two `tasklist` statements.

XML

```
    <group id="WorkItemTracking" description="Workitem definitions
uploading." completionMessage="Work item tracking tasks completed.">
      <dependencies>
        <dependency groupId="Classification" />
        <dependency groupId="Groups" />
      </dependencies>
      <taskList filename="WorkItem Tracking\WorkItems.xml" />
```

```
      <taskList filename="WorkItem Tracking\FabrikamWorkItems.xml" />
    </group>
```

Resolution example

XML

```
    <group id="WorkItemTracking" description="Workitem definitions
uploading." completionMessage="Work item tracking tasks completed.">
      <dependencies>
        <dependency groupId="Classification" />
        <dependency groupId="Groups" />
      </dependencies>
      <taskList filename="WorkItem Tracking\FabrikamWorkItems.xml" />
    </group>
```

TF402575: You can only include one Category definition file in your process.

Error example

The WorkItems.xml file contains two `CATEGORIES` statements.

XML

```
  <task id="Categories" name="Categories definitions"
plugin="Microsoft.ProjectCreationWizard.WorkItemTracking"
completionMessage="Work item type categories created">
    <dependencies>
      <dependency taskId="WITs" />
    </dependencies>
    <taskXml>
      <CATEGORIES fileName="WorkItem Tracking\Categories.xml" />
      <CATEGORIES fileName="WorkItem Tracking\Custom_Categories.xml" />
    </taskXml>
  </task>
```

Resolution example

The WorkItems.xml file has been updated to contain only one `CATEGORIES` statement.

XML

```
  <task id="Categories" name="Categories definitions"
plugin="Microsoft.ProjectCreationWizard.WorkItemTracking"
completionMessage="Work item type categories created">
    <dependencies>
      <dependency taskId="WITs" />
    </dependencies>
    <taskXml>
      <CATEGORIES fileName="WorkItem Tracking\Custom_Categories.xml" />
    </taskXml>
  </task>
```

TF402576: You can only include one ProcessConfiguration definition file in your process.

Error example

The WorkItems.xml file contains duplicate `ProjectConfiguration` statements.

XML

```
    <taskXml>
      <PROCESSCONFIGURATION>
        <ProjectConfiguration fileName="WorkItem
Tracking\Process\ProcessConfiguration.xml"/>
      </PROCESSCONFIGURATION>
    </taskXml>
    <taskXml>
      <PROCESSCONFIGURATION>
        <ProjectConfiguration fileName="WorkItem
Tracking\Process\ProcessConfiguration.xml"/>
      </PROCESSCONFIGURATION>
    </taskXml>
```

Resolution example

The WorkItems.xml file has been updated to contain only one `ProjectConfiguration` statement.

XML

```
    <taskXml>
      <PROCESSCONFIGURATION>
        <ProjectConfiguration fileName="WorkItem
Tracking\Process\ProcessConfiguration.xml"/>
      </PROCESSCONFIGURATION>
    </taskXml>
```

TF402587: Required ProcessConfiguration file missing.

The WorkItems.xml file specifies the ProcessConfiguration file to upload. Either the file isn't specified, contains an out-of-date specification, the specified file is missing from the template, or the folder/file name isn't correct.

Error example

The configuration specified is out-of-date and specifies two files that aren't contained in the Process folder.

XML

```
    <taskXml>
      <PROCESSCONFIGURATION>
        <CommonConfiguration fileName="WorkItem
Tracking\Process\CommonConfiguration.xml"/>
        <AgileConfiguration fileName="WorkItem
Tracking\Process\AgileConfiguration.xml"/>
      </PROCESSCONFIGURATION>
    </taskXml>
```

Resolution example

The WorkItems.xml file has been updated to contain the correct configuration `ProjectConfiguration` statement.

XML

```
    <taskXml>
      <PROCESSCONFIGURATION>
        <ProjectConfiguration fileName="WorkItem
Tracking\Process\ProcessConfiguration.xml"/>
      </PROCESSCONFIGURATION>
    </taskXml>
```

TF402577: Field *[refName]* specifies friendly name *[friendlyName]* which is already in use by fields *[refName]* in processes *[processNames]*.

Friendly names must be unique across all field definitions for all WIT definitions specified for all processes imported to Azure DevOps Services.

The named processes contain WITs that define a field that uses the friendly name.

Modify the WIT definition in your process zip file that contains *[refName]* and specify a different friendly name or make it match an existing field in use.

Error example

The UserStory WIT definition contains `FIELD` element for Fabrikam.Product.Family with friendly name Product.

XML

```
      <FIELD name="Product" refname="Fabrikam.Product.Family" type="String"
reportable="dimension">
        <HELPTEXT>Enter the name of the product family for this story or
feature.</HELPTEXT>
      </FIELD>
```

However, in an existing process, Fabrikam.Product.Versions uses the friendly name *Product*.

XML

```
      <FIELD name="Product" refname="Fabrikam.Product.Versions"
type="String" reportable="dimension">
        <HELPTEXT>Enter the name of the product version for this story or
feature.</HELPTEXT>
      </FIELD>
```

Resolution example

Update the UserStory WIT definition to match the existing field.

XML

```
      <FIELD name="Product" refname="Fabrikam.Product.Versions"
type="String" reportable="dimension">
        <HELPTEXT>Enter the name of the product version for this story or
feature.</HELPTEXT>
      </FIELD>
```

TF402585: Required WorkItemTracking plug-in reference missing from the process template.

You must specify the `Microsoft.ProjectCreationWizard.WorkItemTracking` plug-in in the `metadata` section of the ProcessTemplate.xml file:

Error example

The `Microsoft.ProjectCreationWizard.WorkItemTracking` plug-in is missing from the `plugins` section of the ProcessTemplate.xml file.

XML

```
   <plugins>
      <plugin name="Microsoft.ProjectCreationWizard.Classification"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.Reporting"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.Portal"
wizardPage="true" />
      <plugin name="Microsoft.ProjectCreationWizard.Groups"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.VersionControl"
wizardPage="true" />
      <plugin name="Microsoft.ProjectCreationWizard.TestManagement"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.Build"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.Lab" wizardPage="false"
/>
    </plugins>
```

Resolution example

XML

```
   <plugins>
      <plugin name="Microsoft.ProjectCreationWizard.Classification"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.Reporting"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.Portal"
wizardPage="true" />
      <plugin name="Microsoft.ProjectCreationWizard.Groups"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.WorkItemTracking"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.VersionControl"
```

```
wizardPage="true" />
      <plugin name="Microsoft.ProjectCreationWizard.TestManagement"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.Build"
wizardPage="false" />
      <plugin name="Microsoft.ProjectCreationWizard.Lab" wizardPage="false"
/>
    </plugins>
```

Categories

The Categories.xml definition file must conform to the syntax and rules described in Categories XML element reference.

TF402546: Category *[categoryName]* is missing from the categories file.

All categories that are referenced in the ProcessConfiguration.xml file must be defined in the Categories.xml file. In addition, the system requires that the Categories.xml file contain definitions for the following categories:

- Bug Category
- Code Review Request Category
- Code Review Response Category
- Feedback Request Category
- Feedback Response Category
- Hidden Types Category
- Requirement Category
- Shared Step Category
- Shared Parameter Category
- Task Category
- Test Case Category
- Test Plan Category
- Test Suite Category

Update your Categories.xml file to define the missing *[name]* category.

Error example

ProcessConfiguration references the Epic Category, however it is missing from the Categories.xml definition file.

Resolution example

Epic Category is added to the Categories file.

XML

```
<CATEGORY name="Epic Category" refname="Microsoft.EpicCategory">
  <DEFAULTWORKITEMTYPE name="Epic" />
</CATEGORY>
```

TF402553: Element *[name]* references category *[categoryName]* which isn't defined in the categories file.

Update your Categories.xml file to define the missing *[categoryName]* category.

TF402560: You've defined *[n]* categories. Only *[nn]* are allowed.

Review your Categories.xml file for all `CATEGORY` element statements and determine which ones you can delete so as not exceed the allowed limit. Then, review your ProcessConfiguration.xml file to replace values that reference deleted categories.

TF402596: Category *[categoryName]* doesn't define work Item type *[witName]*.

Categories must reference WITs that are defined in the /WorkItem Tracking/TypeDefinitions folder and listed as a task for upload in the WorkItems.xml plug-in file.
Review your Categories.xml file for references to a WIT that isn't included in the /WorkItem Tracking/TypeDefinitions folder.

Error example

The name of the WIT referenced for Microsoft.EpicCategory is misspelled.

XML

```
<CATEGORY name="Epic Category" refname="Microsoft.EpicCategory">
  <DEFAULTWORKITEMTYPE name="EpicABC" />
</CATEGORY>
```

Resolution example

XML

```
  <CATEGORY name="Epic Category" refname="Microsoft.EpicCategory">
    <DEFAULTWORKITEMTYPE name="Epic" />
  </CATEGORY>
```

TF402597: Custom category *[categoryName]* isn't supported because ProcessConfiguration doesn't reference it.

You can only specify custom categories that you use to configure an Agile tool feature in the ProcessConfiguration.
Review your ProcessConfiguration.xml file and determine if you've missed adding support for a category. Otherwise, remove *[categoryName]* from the Categories.xml file.

Error example

ProcessConfiguration doesn't reference Microsoft.EpicCategory, although it's defined in the Categories.xml definition file.

Resolution example

Add `PortfolioBacklog` to ProcessConfiguration to reference Microsoft.EpicCategory.

XML

```
    <PortfolioBacklog category="Microsoft.EpicCategory" pluralName="Epics"
singularName="Epic" workItemCountLimit="1000">
      <States>
        <State value="New" type="Proposed" />
        <State value="Active" type="InProgress" />
        <State value="Resolved" type="InProgress" />
        <State value="Closed" type="Complete" />
      </States>
      <Columns>
        <Column refname="System.WorkItemType" width="100" />
        <Column refname="System.Title" width="400" />
        <Column refname="System.State" width="100" />
        <Column refname="Microsoft.VSTS.Scheduling.Effort" width="50" />
        <Column refname="Microsoft.VSTS.Common.BusinessValue" width="50" />
        <Column refname="Microsoft.VSTS.CMMI.RequirementType" width="100" />
        <Column refname="System.Tags" width="200" />
      </Columns>
```

```
      <AddPanel>
        <Fields>
          <Field refname="System.Title" />
        </Fields>
      </AddPanel>
    </PortfolioBacklog>
```

Reference articles:

- ProcessConfiguration XML element reference
- Categories XML element reference

Classification plug-in errors

The Classification.xml definition file must conform to the syntax and rules described in Classification plug-in.

TF402511: The Classification plug-in contains duplicate property name: *[propertyName]*.

Error example

Classification.xml file contains a second `property name="MSPROJ"` statement under the `properties` container element.

XML

```
      <properties>
        <property name="MSPROJ" value="Classification\FileMapping.xml"
isFile="true" />
        <property name="MSPROJ"
value="Classification\Fabrikam_FileMapping.xml" isFile="true" />
        <property name="Process Template" value="Fabrikam Agile"/>
      </properties>
```

Resolution example

Remove the duplicate statement.

XML

```
      <properties>
        <property name="MSPROJ"
value="Classification\Fabrikam_FileMapping.xml" isFile="true" />
```

```
        <property name="Process Template" value="Fabrikam Agile"/>
      </properties>
```

TF402512: File *[fileName]* does not exist.

The file specified in the Classification.xml file isn't present in the specified path or the path is misspelled.

Error example

The Classification folder path is misspelled.

XML

```
       <properties>
        <property name="MSPROJ"
value="Classification\Fabrikam_FileMapping.xml" isFile="true" />
        <property name="Process Template" value="Agile"/>
      </properties>
```

Resolution example

XML

```
      <properties>
        <property name="MSPROJ"
value="Classification\Fabrikam_FileMapping.xml" isFile="true" />
        <property name="Process Template" value="Fabrikam Agile"/>
      </properties>
```

TF402513: Name *[pathName]* in the Classification plug-in doesn't conform to TreePath naming restrictions.

Review the `Node` elements you've specified and change the names to conform to supporting naming conventions. Reference: Add and modify area and iteration paths.

Error example

The Path names include the # character which is not allowed.

XML

```
<Node StructureType="ProjectLifecycle" Name="Iteration" xmlns="">
  <Children>
    <Node StructureType="ProjectLifecycle" Name="Sprint #1" />
    <Node StructureType="ProjectLifecycle" Name="Sprint #2" />
    <Node StructureType="ProjectLifecycle" Name="Sprint #3" />
  </Children>
</Node>
```

Resolution example

The Path names have been corrected.

XML

```
<Node StructureType="ProjectLifecycle" Name="Iteration" xmlns="">
  <Children>
    <Node StructureType="ProjectLifecycle" Name="Sprint 1" />
    <Node StructureType="ProjectLifecycle" Name="Sprint 2" />
    <Node StructureType="ProjectLifecycle" Name="Sprint 3" />
  </Children>
</Node>
```

TF402514: Node StructureType *[structureTypeName]* in the Classification plug-in is not supported.

The `StructureType` attribute only allows the following values: `ProjectModelHierarchy` and `ProjectLifecycle`.
Review the `Node` elements that you've specified and remove any unsupported attributes.

Error example

`ProjectLifecycle` has been misspelled.

XML

```
<Node StructureType="ProjectLifecylce" Name="Iteration" xmlns="">
  <Children>
    <Node StructureType="ProjectLifecycle" Name="Sprint 1" />
    <Node StructureType="ProjectLifecycle" Name="Sprint 2" />
    <Node StructureType="ProjectLifecycle" Name="Sprint 3" />
  </Children>
</Node>
```

Resolution example

Misspelled name has been corrected.

XML

```
        <Node StructureType="ProjectLifecycle" Name="Iteration" xmlns="">
          <Children>
            <Node StructureType="ProjectLifecycle" Name="Sprint 1" />
            <Node StructureType="ProjectLifecycle" Name="Sprint 2" />
            <Node StructureType="ProjectLifecycle" Name="Sprint 3" />
          </Children>
        </Node>
```

TF402515: A property attribute in the Classification plug-in is either missing or not supported.

The `property` element only allows the following attributes: `name` , `value` , and `isFile`. Review the `property` elements you've specified and remove any unsupported attributes.

Error example

The `value` attribute is misspelled.

XML

```
       <properties>
        <property name="MSPROJ"
vaule="Classification\Fabrikam_FileMapping.xml" isFile="true" />
        <property name="Process Template" value="Agile" />
      </properties>
```

Resolution example

Misspelled attribute has been corrected.

XML

```
      <properties>
        <property name="MSPROJ"
value="Classification\Fabrikam_FileMapping.xml" isFile="true" />
        <property name="Process Template" value="Fabrikam Agile"/>
      </properties>
```

TF402516: The Classification plug-in contains more than two root level Nodes, which is not supported.

Review the `Node` elements you've specified and remove extra root level nodes.

Error example

Classification.xml file contains a second `Node StructureType="ProjectLifecycle"` statement under the `Nodes` container element.

XML

```
<Nodes>
  <Node StructureType="ProjectLifecycle" Name="Iteration" xmlns="">
    <Children>
      <Node StructureType="ProjectLifecycle" Name="Sprint 1" />
      <Node StructureType="ProjectLifecycle" Name="Sprint 2" />
      <Node StructureType="ProjectLifecycle" Name="Sprint 3" />
    </Children>
  </Node>
  <Node StructureType="ProjectLifecycle" Name="Iteration" xmlns="">
  <Node StructureType="ProjectModelHierarchy" Name="Area" xmlns="" />
</Nodes>
```

Resolution example

Remove the second statement.

XML

```
<Nodes>
  <Node StructureType="ProjectLifecycle" Name="Iteration" xmlns="">
    <Children>
      <Node StructureType="ProjectLifecycle" Name="Sprint 1" />
      <Node StructureType="ProjectLifecycle" Name="Sprint 2" />
      <Node StructureType="ProjectLifecycle" Name="Sprint 3" />
    </Children>
  </Node>
  <Node StructureType="ProjectModelHierarchy" Name="Area" xmlns="" />
</Nodes>
```

TF402523: Area or Iteration path '*[pathName]*' in the GroupsandPermissions plug-is missing from the Classification plug-in.

Add the missing *[pathName]* to the Classification.xml file or remove it from the GroupsandPermissions.xml file.

Error example

Classification.xml file specifies sprints, not iterations.

GroupsandPermissions.xml `teamSettings` specifies Iterations.

XML

```
    <teamSettings areaPath="Area">
        <iterationPaths backlogPath="Iteration">
            <iterationPath path="Iteration 1" />
            <iterationPath path="Iteration 2" />
            <iterationPath path="Iteration 3" />
        </iterationPaths>
    </teamSettings>
```

However, the Classification.xml specifies sprints.

XML

```
        <Node StructureType="ProjectLifecycle" Name="Iteration" xmlns="">
          <Children>
            <Node StructureType="ProjectLifecycle" Name="Sprint 1" />
            <Node StructureType="ProjectLifecycle" Name="Sprint 2" />
            <Node StructureType="ProjectLifecycle" Name="Sprint 3" />
          </Children>
        </Node>
```

Resolution example

Update GroupsandPermissions.xml to use sprints.

XML

```
    <teamSettings areaPath="Area">
        <iterationPaths backlogPath="Iteration">
            <iterationPath path="Sprint 1" />
            <iterationPath path="Sprint 2" />
            <iterationPath path="Sprint 3" />
        </iterationPaths>
    </teamSettings>
```

Link types

> ⓘ Note
>
> Custom link types aren't supported in the current import process.

TF402561: You've defined *[n]* custom link types. Only *[nn]* are allowed.

Review your WorkItems.xml plug-in file for all `LINKTYPE` element statements. Remove statements associated with custom link types so as not exceed the allowed limit. And then, remove the corresponding link type definition file from the LinkTypes folder.

Also, remove any references to the custom link types that you've added to a WIT definition within a `LinksControlOptions` section under the `FORM` section.

TF402583: Custom link type *[name]* is invalid because custom link types aren't supported.

Review your WorkItems.xml plug-in file for all `LINKTYPE` element statements. Remove any statements that specify a custom link type. And then, remove the corresponding link type definition file from the LinkTypes folder.

The following `LINKTYPE` element statements within the WorkItems.xml plug-in file are valid:

XML

```
      <LINKTYPES>
        <LINKTYPE fileName="WorkItem Tracking\LinkTypes\Affects.xml" />
        <LINKTYPE fileName="WorkItem Tracking\LinkTypes\SharedStep.xml" />
        <LINKTYPE fileName="WorkItem Tracking\LinkTypes\TestedBy.xml" />
        <LINKTYPE fileName="WorkItem
Tracking\LinkTypes\SharedParameterLink.xml" />
      </LINKTYPES>
```

Global lists

> ⓘ Note

Global lists aren't supported in the current import process feature.

TF402564: You've defined *[n]* global lists. Only *[nn]* are allowed.

One or more WITs defined in the custom process template contain a `GLOBALLIST` element. Search through your WIT definition files and replace any `GLOBALLIST` elements with `ALLOWEDVALUES` or `SUGGESTEDVALUES` elements. For reference syntax, see Define pick lists.

TF402565: You've defined *[n]* items in global list *[globalListName]*. A maximum of *[nn]* is allowed.

The named `GLOBALLIST` element contains more items than allowed. Either reduce the number of list items contained in the `GLOBALLIST` element, or segment the global list into two or more elements so that neither list exceeds the maximum number of allowed items.

For reference syntax, see Define pick lists.

Process configuration

The ProcessConfiguration.xml definition file must conform to the syntax and rules described in ProcessConfiguration XML element reference.

TF402543: Element *[elementName]* requires that you map exactly one workflow state to metastate *[metastateName]*.

Update the `States` section within the named element in the ProcessConfiguration.xml file to provide the missing metastate mapping or remove extra mappings.

Error example

ProcessConfiguration.xml `RequirementBacklog` element is missing a metastate mapping for `type="Proposed"`.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Backlog items" singularName="Backlog item">
   <States>
      <State value="Committed" type="InProgress" />
      <State value="Done" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

Resolution example

Missing metastate mappings have been added.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Backlog items" singularName="Backlog item">
   <States>
      <State value="New" type="Proposed" />
      <State value="Committed" type="InProgress" />
      <State value="Done" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

TF402547: Element *[elementName]* requires that for work item type *[witName]* you map at least one state to metastate *[metastateName]*.

Review the `WORKFLOW` states defined for the named WIT and then update the `States` section within the named element in the ProcessConfiguration.xml file to provide the missing metastate mapping for the named WIT.

TF402548: Element *[elementName]* requires that you map at least one state to metastate *[metastateName]*.

Update the `States` section within the named element in the ProcessConfiguration.xml file to provide the missing metastate mapping.

Error example

ProcessConfiguration.xml `RequirementBacklog` element is missing a metastate mapping for `type="InProgress"`.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Backlog items" singularName="Backlog item">
   <States>
      <State value="New" type="Proposed" />
      <State value="Done" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

Resolution example

Missing metastate mappings have been added.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Backlog items" singularName="Backlog item">
   <States>
      <State value="New" type="Proposed" />
      <State value="Committed" type="InProgress" />
      <State value="Done" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

TF402550: Element *[elementName]* includes more than one metastate mapping for workflow state *[stateName]*.

Metastate value cannot be mapped to more than one workflow state.

Error example

ProcessConfiguration.xml `RequirementBacklog` element contains two metastate mappings for `value="Active"`.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Stories" singularName="Story">
   <States>
```

```
      <State value="Active" type="Proposed" />
      <State value="Active" type="InProgress" />
      <State value="Resolved" type="InProgress" />
      <State value="Closed" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

Resolution example

Metastate mappings have been corrected.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Stories" singularName="Story">
   <States>
      <State value="New" type="Proposed" />
      <State value="Active" type="InProgress" />
      <State value="Resolved" type="InProgress" />
      <State value="Closed" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

TF402551: Work item type *[witName]* doesn't define workflow state *[stateName]*, which is required because ProcessConfiguration maps it to a metastate for element *[elementName]*.

Either correct the ProcessConfiguration.xml file or the `WORKFLOW` section of the named WIT to add the missing `STATE` and `TRANSITION` elements.

Error example

ProcessConfiguration.xml `RequirementBacklog` element specifies `value="Committed"`, however UserStory.xml doesn't define Committed as a State.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Stories" singularName="Story">
   <States>
      <State value="New" type="Proposed" />
      <State value="Committed" type="InProgress" />
```

```
      <State value="Active" type="InProgress" />
      <State value="Resolved" type="InProgress" />
      <State value="Closed" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

Resolution example

Removed the `State` element for Committed.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Stories" singularName="Story">
   <States>
      <State value="New" type="Proposed" />
      <State value="Active" type="InProgress" />
      <State value="Resolved" type="InProgress" />
      <State value="Closed" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

TF402552: Element *[elementName]* requires that for work item type *[witName]* you map exactly one state to metastate *[metastateName]* for the following states: *[stateNames]*.

Review the `STATES` section in the ProcessConfiguration.xml file for the named element and ensure that each sate listed in the error message is present and mapped to a metastate.

Error example

ProcessConfiguration.xml `RequirementBacklog` element is missing the state `New` which exists on the `User Story` work item type. It should be in the `STATES` list mapped to `type=Proposed"`.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Stories" singularName="Story">
   <States>
```

```
      <State value="Active" type="InProgress" />
      <State value="Resolved" type="InProgress" />
      <State value="Closed" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

Resolution example

Metastate mapping has been corrected.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Stories" singularName="Story">
   <States>
      <State value="New" type="Proposed" />
      <State value="Active" type="InProgress" />
      <State value="Resolved" type="InProgress" />
      <State value="Closed" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

TF402554: Element *[elementName]* specifies an unsupported metastate *[metastateName]*.

Review the `STATES` section in the ProcessConfiguration.xml file for the named element and remove or correct the named metastate.

Error example

ProcessConfiguration.xml `RequirementBacklog` element contains a misspelled metastate mapping for `type=Proposed"`.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Stories" singularName="Story">
   <States>
      <State value="New" type="Propsed" />
      <State value="Active" type="InProgress" />
      <State value="Resolved" type="InProgress" />
      <State value="Closed" type="Complete" />
   </States>
```

```
 . . .
</RequirementBacklog >
```

Resolution example

Metastate mapping has been corrected.

XML

```
<RequirementBacklog category="Microsoft.RequirementCategory"
pluralName="Stories" singularName="Story">
   <States>
      <State value="New" type="Proposed" />
      <State value="Active" type="InProgress" />
      <State value="Resolved" type="InProgress" />
      <State value="Closed" type="Complete" />
   </States>
 . . .
</RequirementBacklog >
```

TF402571: Required element *[elementName]* is missing from Process Configuration.

Edit the ProcessConfiguration.xml file to add the missing named element.

Review ProcessConfiguration XML element reference for required elements.

TF402572: Cannot find specified file in the process template package.

A file specified in a plug-in file for upload is missing from the zip file. Review all files specified for upload and make sure they are included in the process zip file.

Review ProcessConfiguration XML element reference for required elements.

TF402574: ProcessConfiguration doesn't specify required `TypeField` *[typeField]*.

Edit the ProcessConfiguration.xml file to add the missing named `TypeField` element.

Review ProcessConfiguration XML element reference for required `TypeField` elements.

Example

The Scrum process specifies the following `TypeField` elements. If any of these are missing, you'll receive error TF402574.

XML

```
<TypeFields>
    <TypeField refname="System.AreaPath" type="Team" />
    <TypeField refname="Microsoft.VSTS.Scheduling.RemainingWork"
type="RemainingWork" format="format h" />
    <TypeField refname="Microsoft.VSTS.Common.BacklogPriority" type="Order"
/>
    <TypeField refname="Microsoft.VSTS.Scheduling.Effort" type="Effort" />
    <TypeField refname="Microsoft.VSTS.Common.Activity" type="Activity" />
    <TypeField refname="Microsoft.VSTS.Feedback.ApplicationStartInformation"
type="ApplicationStartInformation" />
    <TypeField
refname="Microsoft.VSTS.Feedback.ApplicationLaunchInstructions"
type="ApplicationLaunchInstructions" />
    <TypeField refname="Microsoft.VSTS.Feedback.ApplicationType"
type="ApplicationType">
        <TypeFieldValues>
            <TypeFieldValue value="Web application" type="WebApp" />
            <TypeFieldValue value="Remote machine" type="RemoteMachine" />
            <TypeFieldValue value="Client application" type="ClientApp" />
        </TypeFieldValues>
    </TypeField>
</TypeFields>
```

TF402588: Several portfolio backlogs *[backlogNames]* have defined *[backlogName]* as their parent. A parent backlog may have only one child backlog.

Only one child portfolio backlog can map to a single parent backlog.

Edit ProcessConfiguration to correct the parent-child backlog specifications.

Error example

XML

```
  <PortfolioBacklog category="Microsoft.EpicCategory"
parent="Microsoft.InitiativeCategory" pluralName="Epics"
singularName="Epic">
    ...
  </PortfolioBacklog>
```

```
  <PortfolioBacklog category="Microsoft.FeatureCategory"
parent="Microsoft.InitiativeCategory" pluralName="Features"
singularName="Feature">
    ...
  </PortfolioBacklog>
```

Resolution example

Change the parent on the Feature backlog to point to the Epic backlog.

XML

```
  <PortfolioBacklog category="Microsoft.EpicCategory"
parent="Microsoft.InitiativeCategory" pluralName="Epics"
singularName="Epic">
    ...
  </PortfolioBacklog>

  <PortfolioBacklog category="Microsoft.FeatureCategory"
parent="Microsoft.EpicCategory" pluralName="Features"
singularName="Feature">
    ...
  </PortfolioBacklog>
```

TF402589: Portfolio backlog *[backlogName]* references undefined parent portfolio backlog *[backlogName]*.

The ProcessConfiguration.xml definition contains a `parent` value that references an undefined portfolio backlog.

Error example

XML

```
  <PortfolioBacklog category="Microsoft.FeatureCategory"
parent="Microsoft.EpicCategory" pluralName="Features"
singularName="Feature">
```

Resolution example

Add the Epic `PortfolioBacklog` to the ProcessConfiguration.xml file.

XML

```
  <PortfolioBacklog category="Microsoft.EpicCategory" pluralName="Epics"
singularName="Epic">
    <States>
      <State value="New" type="Proposed" />
      <State value="Active" type="InProgress" />
      <State value="Resolved" type="InProgress" />
      <State value="Closed" type="Complete" />
    </States>
    <Columns>
      <Column refname="System.WorkItemType" width="100" />
      <Column refname="System.Title" width="400" />
      <Column refname="System.State" width="100" />
      <Column refname="Microsoft.VSTS.Common.BusinessValue" width="50" />
      <Column refname="Microsoft.VSTS.CMMI.RequirementType" width="100" />
      <Column refname="System.Tags" width="200" />
    </Columns>
  </PortfolioBacklog>
```

TF402590: Several portfolio backlogs *[Backlog 1, Backlog 2]* don't define their parent.

Only one portfolio backlog, the top backlog, may be unparented. All other backlogs must include a `parent="Microsoft.FooCategory"` attribute and value.

Resolution example

XML

```
  <PortfolioBacklog category="Microsoft.FeatureCategory"
parent="Microsoft.EpicCategory" pluralName="Features"
singularName="Feature">
```

402595: Too many portfolio backlogs are defined. A maximum of 5 are allowed.

Edit the ProcessConfiguration.xml file to remove the extra `PortfolioBacklog` elements from the `PortfolioBacklogs` section.

Review ProcessConfiguration XML element reference for more information.

TF400506: This element defines the states for work items that represent Bugs or Defects.

This error occurs when the `BugWorkItems` section is misconfigured. Each state defined for a work item type included in the `Microsoft.BugCategory` must exist in the set of State values defined in the `BugWorkItems` section. For example, if the following states are defined for work item types defined for Bug work item type—*New, Approved, Committed, Done*—then the `BugWorkItems` section should be configured as follows:

XML

```
  <BugWorkItems category="Microsoft.BugCategory" pluralName="Bugs"
singularName="Bug">
    <States>
    <States>
      <State value="New" type="Proposed" />
      <State value="Approved" type="Proposed" />
      <State value="Committed" type="InProgress" />
      <State value="Done" type="Complete" />
    </States>
  </BugWorkItems>
```

For more information, see ProcessConfiguration XML element reference.

TF400507: Each work item type must support an initial state value that matches one of the states defined in: BugWorkItems.

This error occurs when a work item type included in the `Microsoft.BugCategory` specifies an initial state that isn't included in the `BugWorkItems` section of the ProcessConfiguration XML file. Either fix the XML definition workflow states for the named work item types, or update the `BugWorkItems` section to include the missing state(s).

For more information, see ProcessConfiguration XML element reference.

TF400572: The Project Process Settings must be configured for this feature to be used.

This is caused by templates that are from TFS 2010 or earlier versions, before the Project Process Settings existed. Try running Configure features after an upgrade to resolve, or add the ProcessConfiguration XML file to the process template.

Work item types

TF402570: You've defined *[n]* work item types. A maximum of *[witLimit]* is allowed.

You've defined more WITs that are allowed in the process. Review your WorkItems.xml file and reduce the number of `WORKITEMTYPE` statements it contains and remove the associated WIT definition files from the process.

TF402573: Work item type WIT doesn't specify required `refname` attribute.

Work item types (Bug, User Story, Task, etc.) require the `refname` attribute. The refname value must be unique and cannot contain any reserved namespaces. The namespaces-System.*XXX* and Microsoft.VSTS.*XXX*-are reserved by Azure DevOps.

See All WITD XML elements reference for more information.

Error example

XML

```
<WORKITEMTYPE name="Bug">
```

Resolution example

XML

```
<WORKITEMTYPE name="Bug" refname="MyCompany.Bug">
```

TF402599: The work item type refname *[refName]* isn't valid as it uses a disallowed namespace [namespace].

Reference names of custom fields and WITs can't use reserved namespaces: System.*XXX* and Microsoft.VSTS.*XXX*.

Edit the `refname` attribute of the named WIT.

Error example

XML

```
<FIELD name="Custom Field" refname="Microsoft.VSTS.CustomField"
type="String" />
```

Resolution example

XML

```
<FIELD name="Custom Field" refname="*CustomNamespace.CustomField*"
type="String" />
```

TF402600: The work item type reference name *[refName]* isn't valid. Work item type reference names must contain only letters, no spaces, and at least one period (.)

WIT reference names must adhere to established naming conventions: only letters, no spaces, and at least one period (.)

Edit the `refname` attribute of the named WIT to meet naming requirements..

Error example

XML

```
<WORKITEMTYPE name="Bug" refname="MyCompanyBug32">
```

Resolution example

XML

```
<WORKITEMTYPE name="Bug" refname="MyCompany.Bug">
```

Work item fields

TF402538: Field rule *[ruleName]* isn't supported.

Edit your WIT definitions to remove the named rule. The following rules are **not** supported for import to Azure DevOps: `MATCH`, `CANNOTLOSEVALUE`, `PROHIBITEDVALUES`, and `NOTSAMEAS`.

TF402539: Field *[refName]* only allows the following rules: *[ruleNames]*.

Edit your WIT definitions for the named field and remove any non-allowed rule(s). Review both the `FIELDS` and `WORKFLOW` sections. Most system fields do not allow specifying rules.

Valid rules to use with System fields

Name	Allowed rules
System.Title	`REQUIRED`, `DEFAULT`
System.Description	`REQUIRED`, `DEFAULT`
System.AssignedTo	`REQUIRED`, `DEFAULT`, `ALLOWEXISTINGVALUE`, `VALIDUSER`
System.ChangedBy	`REQUIRED`, `DEFAULT`, `ALLOWEXISTINGVALUE`, `VALIDUSER`

TF402540: Work item type *[witName]* isn't valid because it specifies global list *[GLOBALLIST]*. Global lists aren't supported.

Global lists are not supported in VSTS. Replace all instances of `GLOBALLIST` elements with `ALLOWEDVALUES` and `LISTITEM` elements in all WIT definition files. Reference: Define pick lists.

Error example

XML

```
<FIELD name="CustomField" refname="MyCompany.CustomField" type="String">
  <ALLOWEDVALUES>
    <GLOBALLIST name="Disciplines" />
  </ALLOWEDVALUES>
</FIELD>
```

Resolution example

XML

```
<FIELD name="CustomField" refname="MyCompany.CustomField" type="String">
  <ALLOWEDVALUES>
    <LISTITEM value="Architecture" />
    <LISTITEM value="Requirements" />
    <LISTITEM value="Development" />
    <LISTITEM value="Release Management" />
    <LISTITEM value="Project Management" />
    <LISTITEM value="Test" />
  </ALLOWEDVALUES>
</FIELD>
```

TF402541: Work item type *[witName]* isn't valid because it references global list *[globalListName]*. Global lists aren't supported.

Global lists are not supported in VSTS. Replace all instances of `GLOBALLIST` elements with `ALLOWEDVALUES` and `LISTITEM` elements in all WIT definition files. Reference: Define pick lists.

Error example

XML

```
<FIELD name="CustomField" refname="MyCompany.CustomField" type="String">
  <ALLOWEDVALUES>
    <GLOBALLIST name="Disciplines" />
  </ALLOWEDVALUES>
</FIELD>
```

Resolution example

XML

```
<FIELD name="CustomField" refname="MyCompany.CustomField" type="String">
  <ALLOWEDVALUES>
    <LISTITEM value="Architecture" />
    <LISTITEM value="Requirements" />
    <LISTITEM value="Development" />
    <LISTITEM value="Release Management" />
    <LISTITEM value="Project Management" />
    <LISTITEM value="Test" />
  </ALLOWEDVALUES>
</FIELD>
```

TF402542: The custom field refname *[refName]* isn't valid as it uses disallowed namespace *[namespace]*.

The namespaces-System.*XXX* and Microsoft.VSTS.*XXX*-are reserved in VSTS. Reference names of custom fields and types can't use these namespaces.

To fix this error, rename the `refname` attribute for the named field in the WIT definition files where it appears.

Error example

XML

```
<FIELD name="CustomField" refname="System.CustomField" type="String" />

  - OR -

<FIELD name="CustomField" refname="Microsoft.VSTS.CustomField" type="String"
/>
```

Resolution example

XML

```
<FIELD name="CustomField" refname="MyCompany.CustomField" type="String" />
```

TF402544: Field *[refName]*, defined in work item type *[witName]*, requires an `ALLOWEDVALUES` rule that contains values to support element *[elementName]* specified in ProcessConfiguration.

Edit the named field in the named WIT to provide the missing `ALLOWEDVALUES` rule, referencing the named element for more details.

The ProcessConfiguration.xml file element, `TypeField refname="Microsoft.VSTS.Common.Activity" type="Activity"`, requires a pick list is defined for the Microsoft.VSTS.Common.Activity field in all WITs that belong to the Task Category.

XML

```
<TypeField refname="Microsoft.VSTS.Common.Activity" type="Activity" />
```

If you have added bugs to the Task Category, you must ensure that the Bug.xml file contains the named field and specifies the same pick list as is defined for it in the Task.xml file.

Error example

Bug.xml has the field defined, but not the pick list.

XML

```
      <FIELD name="Activity" refname="Microsoft.VSTS.Common.Activity"
type="String" reportable="dimension">
        <HELPTEXT>Type of work involved</HELPTEXT>
      </FIELD>
```

Resolution example

Corrected Bug.xml

XML

```
      <FIELD name="Activity" refname="Microsoft.VSTS.Common.Activity"
type="String" reportable="dimension">
        <HELPTEXT>Type of work involved</HELPTEXT>
        <SUGGESTEDVALUES>
          <LISTITEM value="Development"/>
          <LISTITEM value="Testing"/>
          <LISTITEM value="Requirements"/>
          <LISTITEM value="Design"/>
          <LISTITEM value="Deployment"/>
          <LISTITEM value="Documentation"/>
        </SUGGESTEDVALUES>
      </FIELD>
```

TF402545: Element *[elementName]* requires that you include field *[fieldName]* in the definition of work item type *[witName]*.

You should include a corresponding `FIELD` element in the named WIT definition for each `field` that you specify within the `AddPanel` section of the

ProcessConfiguration.xml file.
You should also include a `Control` element within the `FORM` section of the named WIT.

Error example

ProcessConfiguration.xml specifies two custom fields. However, these fields aren't defined in the UserStory.xml file.

XML

```
<AddPanel>
   <Fields>
      <Field refname="System.Title" />
      <Field refname="Fabrikam.Product" />
      <Field refname="Fabrikam.Technology" />
   </Fields>
</AddPanel>
```

Resolution example

Missing `FIELD` elements added to the UserStory.xml file.

XML

```
      <FIELD name="Product" refname="Fabrikam.Product" type="String"
reportable="dimension">
     <FIELD name="Technology" refname="Fabrikam.Technology" type="String"
reportable="dimension">
```

To support ProcessConfiguration element *[elementName]*, work item type *[witName]* must define TypeField *[typeField]* (field refname [refName]).

In the ProcessConfiguration you created a `TypeField` with `TypeFieldValues`. In your work item type you have to reference that same `TypeField`.

Error example

ProcessConfiguration.xml

XML

```
  <TypeField refname="Custom.ApplicationType" type="ApplicationType">
    <TypeFieldValues>
      <TypeFieldValue value="Web application" type="WebApp" />
      <TypeFieldValue value="Remote machine" type="RemoteMachine" />
      <TypeFieldValue value="Client application" type="ClientApp" />
    </TypeFieldValues>
  </TypeField>
```

FeedbackRequest.xml is using the Microsoft.VSTS.Feedback.ApplicationType field when it should be using the Custom.ApplicationType field.

XML

```
  <FIELD name="Application Type"
refname="Microsoft.VSTS.Feedback.ApplicationType" type="String">
    ...
  </FIELD>
```

Resolution example

FeedbackRequest.xml

XML

```
  <FIELD name="Application Type" refname="Custom.ApplicationType"
type="String">
    ...
  </FIELD>
```

TF402556: For field *[refName]* to be well defined, you must name it *[fieldName]* and set its type to *[fieldType]*. Provided *[refName]* is *[fieldName]* and type is *[fieldType]*.

If you are on an on-premises server and running **tfsMigrator**, this error is usually generated when your process is out-of-date and a system field is not properly defined. In the majority of those cases you need to use the `witadmin /changefield` command to properly rename the field.

Examples of Reserved Fields

Reserved System.*XXX* and Microsoft.VSTS.*XXX* fields have required name and type values.

Field	Name	Type
System.Id	ID	Integer
System.Title	Title	String
Microsoft.VSTS.Scheduling.StoryPoints	Story Points	Double
Microsoft.VSTS.Scheduling.RemainingWork	Remaining Work	Double
Microsoft.VSTS.Scheduling.OriginalEstimate	Original Estimate	Double
Microsoft.VSTS.Scheduling.CompletedWork	Completed Work	Double

Error example

TF402556: For field Microsoft.VSTS.TCM.ReproSteps to be well defined, you must name it Repro Steps and set its type to HTML. Provided Microsoft.VSTS.TCM.ReproSteps is My Repro Steps and type is HTML.

In Bug.xml, the friendly field name has been changed to "My Repro Steps".

XML

```
  <FIELD name="My Repro Steps" refname="Microsoft.VSTS.TCM.ReproSteps"
type="HTML">
    <HELPTEXT>How to see the bug. End by contrasting expected with actual
behavior.</HELPTEXT>
  </FIELD>
```

Resolution example

Bug.xml

XML

```
  <FIELD name="Repro Steps" refname="Microsoft.VSTS.TCM.ReproSteps"
type="HTML">
    <HELPTEXT>How to see the bug. End by contrasting expected with actual
behavior.</HELPTEXT>
  </FIELD>
```

TF402557: Inconsistent definitions exist for field *[refName]* in the following work item types: *[witName]*.

Ensure that all references to a field have the same *RefName*, *Name*, and *Type*.

`FIELD` element definitions for fields with the same friendly name must be consistent across WITs. Check to ensure that the `refname`, `name`, and `type` attributes are the same in each of your WIT definitions for the named field.

Error example

Bug.xml

XML

```
  <FIELD name="Hair Color" refname="MyCompany.CustomFields.HairColor"
type="String" reportable="dimension" />
```

UserStory.xml

XML

```
  <FIELD name="Hair Color 2" refname="MyCompany.CustomFields.HairColor"
type="Double" reportable="dimension" />
```

Notice that the `name` and `type` attributes differ from that in the Bug.xml work item type

Resolution example

Bug.xml

XML

```
  <FIELD name="Hair Color" refname="MyCompany.CustomFields.HairColor"
type="String" reportable="dimension" />
```

UserStory.xml

XML

```
  <FIELD name="Hair Color" refname="MyCompany.CustomFields.HairColor"
type="String" reportable="dimension" />
```

TF402558: The definition of field *[refName]* is inconsistent with the existing definition of the field in the following templates: *[templateNames]*.

Fields must be defined consistently across processes that have been or that you plan to import to Azure DevOps Services. The system doesn't allow you to import a process that contains a `FIELD` element which is defined one way in Process A and another way in Process B. All `FIELD` element attributes must match.

Example

Process A, Bug.xml

XML

```
  <FIELDS>
    ...
    <FIELD name="Foo" refname="MyCompany.CustomFields.Foo" type="String"
reportable="dimension" />
    ...
  </FIELDS>
```

Process B, Bug.xml

XML

```
  <FIELDS>
    ...
    <FIELD name="Bar" refname="MyCompany.CustomFields.Foo" type="Double"
reportable="dimension" />
    ...
  </FIELDS>
```

Notice how different values are specified for `name` and `type` attributes.

Resolution

Ensure that fields of the same `refname` are defined the same across all processes that have been imported or that you plan to import to Azure DevOps Services.

TF402559: The definition for field *[refName]* is inconsistent with an existing field. The `type` is

[typeName] but should be *[typeName]*.

Edit the WIT file(s) that contains the named field to ensure the specified `type` attribute value is consistent across all WITs.

TF402562: You've defined *[n]* fields for work item type *[witName]*. Only *[fieldLimit]* are allowed.

You've defined *[n]* fields for work item type *[witName]*. Only *[fieldLimit]* are allowed. Within a single WIT, you can specify only the specified number of fields.

Edit the named WIT and remove the extra custom fields to reduce the total number of fields to be within the allowed limit.

TF402563: You've defined a total of *[n]* fields for all work item types. Only *[fieldLimit]* are allowed.

You've defined *[n]* fields for all WITs defined in the process. Only *[fieldLimit]* are allowed. This includes System.*XXX* and Microsoft.VSTS.*XXX* namespace fields as well as custom fields.

Review the `FIELDS` and `WORKFLOW` sections and determine which custom fields to remove.

TF402566: You've defined *[n]* field rules for work item type *[witName]*. A maximum of *[ruleLimit]* is allowed.

You've defined *[n]* field rules for the named WIT which exceeds the allowed number of field rules for any one WIT.

Edit the named WIT and remove the extra field rules to reduce the total number of fields to be within the allowed limit.

TF402568: You've defined *[n]* fields with `syncnamechanges="true"` for work item type *[witName]*. A maximum of 64 is allowed.

The number of fields with `syncnamechanges="true"` defined for the named WIT exceeds the allowed limit. This includes System.*XXX* and Microsoft.VSTS.*XXX* namespace fields as well as custom fields.

Review the `FIELDS` section of the named WIT and determine which custom fields to remove or modify.

Example

XML

```
  <FIELD name="Assigned To" refname="System.AssignedTo" type="String"
reportable="dimension" syncnamechanges="true" >
    ...
  </FIELD>
```

TF402569: You've defined *[n]* values for field *[fieldName]* in work item type *[witName]*. A maximum of *[listLimit]* is allowed.

The number of `LISTITEM` elements defined for the named field in the named WIT exceeds the allowed limit.

Edit the named WIT to reduce the number of `LISTITEM` elements so as not to exceed the allowed maximum.

Example

XML

```
  <FIELD name="Favorite Color" refname="MyCompany.CustomFields.FavColor"
type="String" reportable="dimension">
    <ALLOWEDVALUES>
      <LISTITEM value="Color1" />
      <LISTITEM value="Color2" />
      <LISTITEM value="Color3" />
      . . .
    </ALLOWEDVALUES>
  </FIELD>
```

Notice the bug and user story `refname` values are different.

TF402584: Field rule attributes "for" or "not" aren't supported.

The `"for"` and `"not"` attributes are not supported at all for any field rule for import to Azure DevOps Services.
Review the `FIELDS` and `WORKFLOW` sections for the presence of `"for"` and `"not"` attributes and remove them.

Error example

XML

```xml
<FIELD name="Title">
  <READONLY for="Dev Team" not="Test Team" />
</FIELD>
```

Resolution example

XML

```xml
<FIELD name="Title">
  <READONLY />
</FIELD>
```

TF402593: Field rules aren't supported for field *[fieldName]*.

Most System and Microsoft.VSTS fields do not support rules. See Rules and rule evaluation for a complete list of supported field rules.

Edit the definition files for work item types that contain the named field and remove the field rules specified for it.

TF402602: The field *[refName]* must have exactly these values *[values]*.

All fields defined by VSTS in the reserved namespaces-System.*XXX* and Microsoft.VSTS.*XXX*-must not be altered. If you wish to have a different list of values in these pick list fields, define your own custom field.

Error example

Bug.xml has the priority field defined, but has a different list of values than expected.

XML

```
      <FIELD name="Priority" refname="Microsoft.VSTS.Common.Priority"
type="Integer" reportable="dimension">
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="0"/>
          <LISTITEM value="1"/>
          <LISTITEM value="2"/>
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="2" />
      </FIELD>
```

Resolution example

Corrected Bug.xml including a new field

XML

```
      <FIELD name="Priority" refname="Microsoft.VSTS.Common.Priority"
type="Integer" reportable="dimension">
        <HELPTEXT>Business importance. 1=must fix; 4=unimportant.</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="1"/>
          <LISTITEM value="2"/>
          <LISTITEM value="3"/>
          <LISTITEM value="4"/>
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="2" />
      </FIELD>

      <FIELD name="Custom Priority" refname="Custom.Priority" type="Integer"
reportable="dimension">
        <HELPTEXT>Business importance. 1=must fix; 4=unimportant.</HELPTEXT>
        <ALLOWEDVALUES expanditems="true">
          <LISTITEM value="0"/>
          <LISTITEM value="1"/>
          <LISTITEM value="2"/>
        </ALLOWEDVALUES>
        <DEFAULT from="value" value="2" />
      </FIELD>
```

VS402504: User or group cannot be found *[project]\Group Name*. Verify that the users and groups used in your work item type definition exist.

Custom project scoped groups are not supported. You can only reference account level scoped groups.

Error example

Bug.xml is referencing a project scoped group.

```
... "[project]\Organization Leaders"
```

Resolution example

Create a new account (collection) level group "Organization Leaders" and reference it accordingly in the XML.

```
... "[global]\Organization Leaders"
```

> ⓘ **Note**
>
> A freshly created global group might take a little while before being found by the import process.

TF237094: Field name '[fieldName]' is used by the field '[refName]', so it cannot be used by the field '[refName]'.

Field name already exist with that same name on a different field. Change the name of the field.

Error example

`TF237094: Field name 'External ID' is used by the field 'Custom.ExistingField.ExternalID', so it cannot be used by the field 'Custom.NewField.ExternalID'.`

Resolution example

Change the field name of Custom.NewField.ExternalID

```
XML
```

```
<FIELD name="External ID" refname="Custom.NewField.ExternalID" type="string"
reportable="dimension" />
```

Work item layout

VS403104: Work item type *[witName]* references the required field *[refName]* which is not included in all layouts.

This warning is generated if you have a required field in a work item type that is referenced in the `<Layout>` node but not the `<WebLayout>`. The `<WebLayout>` is used to modify the layout of the new form. See WebLayout xml reference for details.

VS403073: Group & `<Group Name>`: violates the rule that a group can only contain a single HTML or WebPage control preceded by label controls..

In the new form layout, a group can only contain one HTMLFieldControl or WebPageControl.

Error example

XML

```
<Section>
    <Group Label="Description:">
        <Control Label="Reason For Request:" Type="HtmlFieldControl"
FieldName="System.Description" />
        <Control Label="Business Case For Request:" Type="HtmlFieldControl"
FieldName="MB.BusinessCase" />
    </Group>
</Section>
```

To resolve this, create two separate groups that contain one control each.

Resolution example

XML

```
<Section>
    <Group Label="Reason for Request">
        <Control Label="Reason For Request:" Type="HtmlFieldControl"
FieldName="System.Description" />
    </Group>
    <Group Label="Business Case">
        <Control Label="Business Case For Request" Type="HtmlFieldControl"
FieldName="Custom.BusinessCase" />
    </Group>
</Section>
```

TF402594: File violates the schema with the following error: The element 'Control' cannot contain child element 'Link' because the parent element's content model is empty.

Error example

XML

```
  <Control Type="FieldControl" FieldName="customfield.foo" Label="Foo"
LabelPosition="Left">
    <Link UrlRoot="http://www.visualstudio.microsoft.com/team-services/" />
  </Control>
```

Resolution example

XML

```
  <Control Type="FieldControl" FieldName="System.Title" LabelPosition="Left"
Label="Title 1"
    <LabelText>
        <Text>
          <Link UrlRoot="http://www.visualstudio.microsoft.com/team-
services/" />
              Title 2
        </Text>
    </LabelText>
  </Control>
```

Workflow definitions

TF402567: You've defined *[n]* workflow states for work item type *[witName]*. A maximum of *[stateLimit]* is allowed.

You've defined *[n]* workflow states for the named WIT which exceeds the allowed number of workflow states for any one WIT.

Edit the `WORKFLOW` section of the named WIT and remove the extra `STATE` elements to reduce the total number of states to be within the allowed limit.

TF402578: Field *[refName]* specifies friendly name *[friendlyName]* which is already in use by field *[refName]*. Friendly names must be unique across all field definitions.

Field names must be unique within the work item type.

Error example

XML

```
  <FIELD name="Foo" refname="MyCompany.CustomFields.Foo" type="String"
reportable="dimension" />
  <FIELD name="Foo" refname="MyCompany.CustomFields.Bar" type="String"
reportable="dimension" />
```

Notice there are two fields with the name `<FIELD name="Foo"`

Resolution example

XML

```
  <FIELD name="Foo" refname="MyCompany.CustomFields.Foo" type="String"
reportable="dimension" />
  <FIELD name="Bar" refname="MyCompany.CustomFields.Bar" type="String"
reportable="dimension" />
```

TF402579: Name *[friendlyName]* is used multiple times on different fields across the following work item types: *[witNames]*. Name for fields must be unique across the project collection.

Fields referencing the same `refname="MyCompany.FieldName"` must have the same friendly name `FIELD name="fieldname"` value.

Error example

Process A, Bug.xml

```xml
  <FIELD name="Foo" refname="MyCompany.CustomFields.Foo" type="String"
reportable="dimension" />
```

Process B, Bug.xml

```xml
  <FIELD name="Bar" refname="MyCompany.CustomFields.Foo" type="String"
reportable="dimension" />
```

Because both fields share the same reference name, `refname="MyCompany.CustomFields.Foo"`, they must also specify the same friendly name `<FIELD name="Foo"` across all processes that have been imported or will be imported to Azure DevOps Services.

TF402580: You can only use the name *[witName]* for a single work item type.

WIT friendly names `WORKITEMTYPE name="Name"` must be unique within the process.

Error example

My Work Item A.xml

```xml
  <WORKITEMTYPE name="My Work Item" refname="My.MyWorkItemA">
```

My Work Item B.xml

```xml
  <WORKITEMTYPE name="My Work Item" refname="My.MyWorkItemB">
```

Notice how the `WORKITEMTYPE name=` is the same across both work item types.

Resolution example

My Work Item A.xml

```XML
<WORKITEMTYPE name="My Work Item A" refname="My.MyWorkItemA">
```

My Work Item B.xml

```XML
<WORKITEMTYPE name="My Work Item B" refname="My.MyWorkItemB">
```

TF402581: You can only use the refname *[refName]* for a single work item type.

WIT reference names `refname="value"` must be unique within the process.

Error example

Bug.xml

```XML
<WORKITEMTYPE name="Bug" refname="MyCompany.Name">
```

UserStory.xml

```XML
<WORKITEMTYPE name="User Story" refname="MyCompany.Name">
```

notice both `refname` values equal "MyCompany.Name"

Resolution example

Bug.xml

XML

```
  <WORKITEMTYPE name="Bug" refname="MyCompany.Bug">
```

UserStory.xml

XML

```
  <WORKITEMTYPE name="User Story" refname="MyCompany.UserStory">
```

TF402582: Work item type *[witName]* contains custom control *[controlName]* which is not supported.

Custom controls are not supported in Azure DevOps Services. Review the `FORM` section for the named WIT and remove any custom controls that are defined.

Error example

`Type="OneViewMultiValueControl"` specifies a custom control. This must be removed or replaced to a supported control.

XML

```
    <Group Label="Engineering Alignment">
        <Column PercentWidth="100">
            <Control FieldName="Fabrikam.Content.Product"
Type="FieldControl" Label="Product" LabelPosition="Left" />
            <Control FieldName="Fabrikam.Content.Release"
Type="FieldControl" Label="Milestone" LabelPosition="Left" />
            <Control FieldName="Fabrikam.Content.Technology"
Type="FabrikamMultiValueControl" Label="Technology" LabelPosition="Left" />
        </Column>
    </Group>
```

Test management plug-in errors

The TestManagement.xml file, located in the Test Management folder, must conform to the syntax and rules described in Define the initial configuration of Test Manager.

TF402533: Element '*[elementName]*' in TestManagement plug-in is missing the `'fileName'` attribute.

Add the missing attribute to the named element in the TestManagement.xml file.

TF402534: TestManagement plug-in contains an unsupported task '*[taskName]*'

Correct the named tasked in your TestManagement.xml file.

Error example

The `TestResolutionStates` element has been misspelled.

XML

```
    <taskXml>
      <TestResolutionSattes fileName="Test
Management\TestResolutionState.xml" />
    </taskXml>
```

Resolution example

Corrected misspellings.

XML

```
    <taskXml>
      <TestResolutionStates fileName="Test
Management\TestResolutionState.xml" />
    </taskXml>
```

Related articles

- Validate and import processes
- Azure DevOps Data Import Service

Supported operations when moving from Hosted XML to an inherited process

Article • 02/24/2023

Azure DevOps Services

Upgrading a Hosted XML process model to an inherited process provides the convenience of customizing your work tracking system through the user interface. For an overview of supported customizations available to you with the Inheritance process, see About process customization and inherited processes.

While the clone process attempts to model all your work tracking customizations, there are some limitations. This article outlines the set of customizations that are supported during the clone process and those which aren't.

The Inheritance process model supports most customizations, however some of the more advanced customizations you made with the Hosted XML process might not be supported. In addition, some of the customizations made to the Hosted XML process need to be manually created in the inherited process.

> ⓘ **Note**
>
> Before you **change the process of an existing project** from Hosted XML to the cloned inherited process, review this article to understand which customizations are preserved and which are ignored.

Customizations preserved during clone

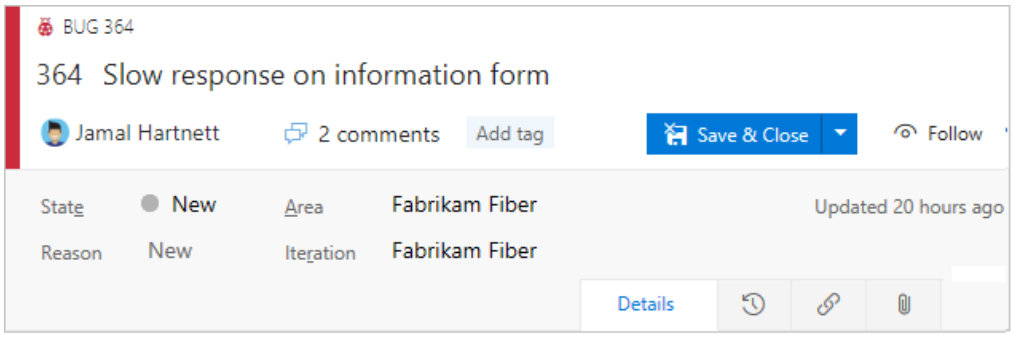
When you clone a Hosted XML process to an inherited process, the customizations listed in the following table are preserved.

Artifact	Description
Work item types (WITs)	All system and custom WITs are preserved. Customizations made to WIT color and icon are preserved.

Artifact	Description
Work item fields	All custom fields are preserved. Fields that reference global lists are updated with picklists. All default values are ignored. To learn more about supported field customizations, see About process customization and inherited processes, Field customizations.
Workflow states	All system and custom workflow states are preserved.
Workflow state categories	All customizations made to the ProcessConfiguration XML file to map a workflow state to a state category (*Proposed, In Progress, Resolved, Completed*) are preserved. Only one workflow state can be assigned to the *Completed* state category. If you have assigned a custom workflow state to the *Completed* state category, it is preserved upon clone. Any workflow state for a work item type that isn't included in a backlog level gets assigned to the *In Progress* state category. Check all custom workflow states post clone. To learn more, see Workflow states and state categories.
Work item form layout	A best effort is made to preserve the customizations made to the web form layout. However, any customizations made to the header area are ignored. Specifically, the **Weblayout** `ShowEmptyReadOnlyFields` attribute assignment is ignored.
Backlog levels	Additions and customizations made to the product backlog and portfolio backlog levels are preserved.
Global lists	Global lists are converted to picklists for individual fields.
Default properties	The default properties set for teams that you add to a project are preserved as documented in Process configuration XML element reference, Specify properties and behaviors.

Customizations ignored during clone

Artifact	Description

Artifact	Description
Header area customization	Any customizations made to the header area within the work item form are ignored. The header area, as shown in the following image, is managed by the system. Any customizations made within the **SystemControls** section of the **WebLayout** are ignored. 
Four column layout and size	The inherited process supports a a fixed relative sizing of three columns to a WIT layout, while the Hosted XML process supports up to four columns and allows you to set the first column as equal sized with the rest of the columns.
Hide Details page in layout	The inherited process ignores any customizations made to hide the Details page in a WIT layout.
Workflow restriction	The inherited process follows an any-to-any workflow state transition. Any customizations that restrict the transition from one workflow state to another are ignored.
Workflow state reasons	Customized reasons added to workflow states are ignored.
Conditional picklists	Conditional picklists, also referred to as dependent or cascading picklists, are ignored. Multiple sets of allowed values per field are ignored. Picklists are defined for a field at the collection level and shared across processes and WITs.
Custom rules	All custom rules to fields and workflow are ignored.
Custom link controls	Custom link controls are ignored.
Extensions	The inherited process supports an opt-out model for custom control extensions, while the Hosted XML process supports an opt-in model. This means that work item types defined within the cloned inherited process show all contributions from all installed and enabled extensions. You can selectively hide or remove them as needed.
Categories	Changes made to a default category are preserved, but any custom categories are ignored. Also note that system work item types such as Issue or Impediment are not supported on a backlog level.

Artifact	Description
Identity fields with string values	Lists that contain an identity value in ALLOWEDVALUES or PROHIBITEDVALUES are automatically converted into the Identity field type. Any other string values in the list are ignored.

Post-upgrade customizations to make manually

The upgrade makes a best-effort attempt to reconcile the system process and the customizations made to the Hosted XML process. After you upgrade, we recommend you review the inherited process and reapply customizations manually.

- Create a test project: Use to verify the customizations preserved or reapplied to a process
- Update the default value for any field: define any default values you had previously defined
- Workflow states: verify the mapping of states to workflow state categories
- Custom rules: You can recreate select rules as needed. Rules for the Hosted XML process model don't map one-to-one to rules defined for an Inherited process. Specifically:
  - Several rules are already defined in the system process or autogenerated. For example, certain system fields such as Changed By, Change Date, Closed By, Closed Date are governed by system rules.
  - Some rules are now specified as field attributes, such as making a field a default or required.
- Disable work item types.
- Hide inherited fields or controls.
- Custom controls: verify that custom controls are applied as expected; disable or hide unwanted groups or page extensions.

Related articles

- About process customization and inherited processes
- Clone a Hosted XML process to Inheritance
- Change a project from a Hosted XML process to Inheritance

Clone a Hosted XML process to an Inheritance process

Article • 02/24/2023

Azure DevOps Services

Cloning your Hosted XML process to the Inheritance process model allows you to manage customizations through the user interface versus XML definition files. The inherited process supports the set of customizations described here About process customization and inherited processes.

Before you clone a Hosted XML process, we recommend that you review Supported operations when moving from Hosted XML process to inherited process.

Prerequisites

- Your account currently uses the Hosted XML process model
- You must be a member of the Project Collection Administrators group, the organization owner, or be granted explicit permissions to edit, create, or manage a specific process and to create a project.

Sequence of steps

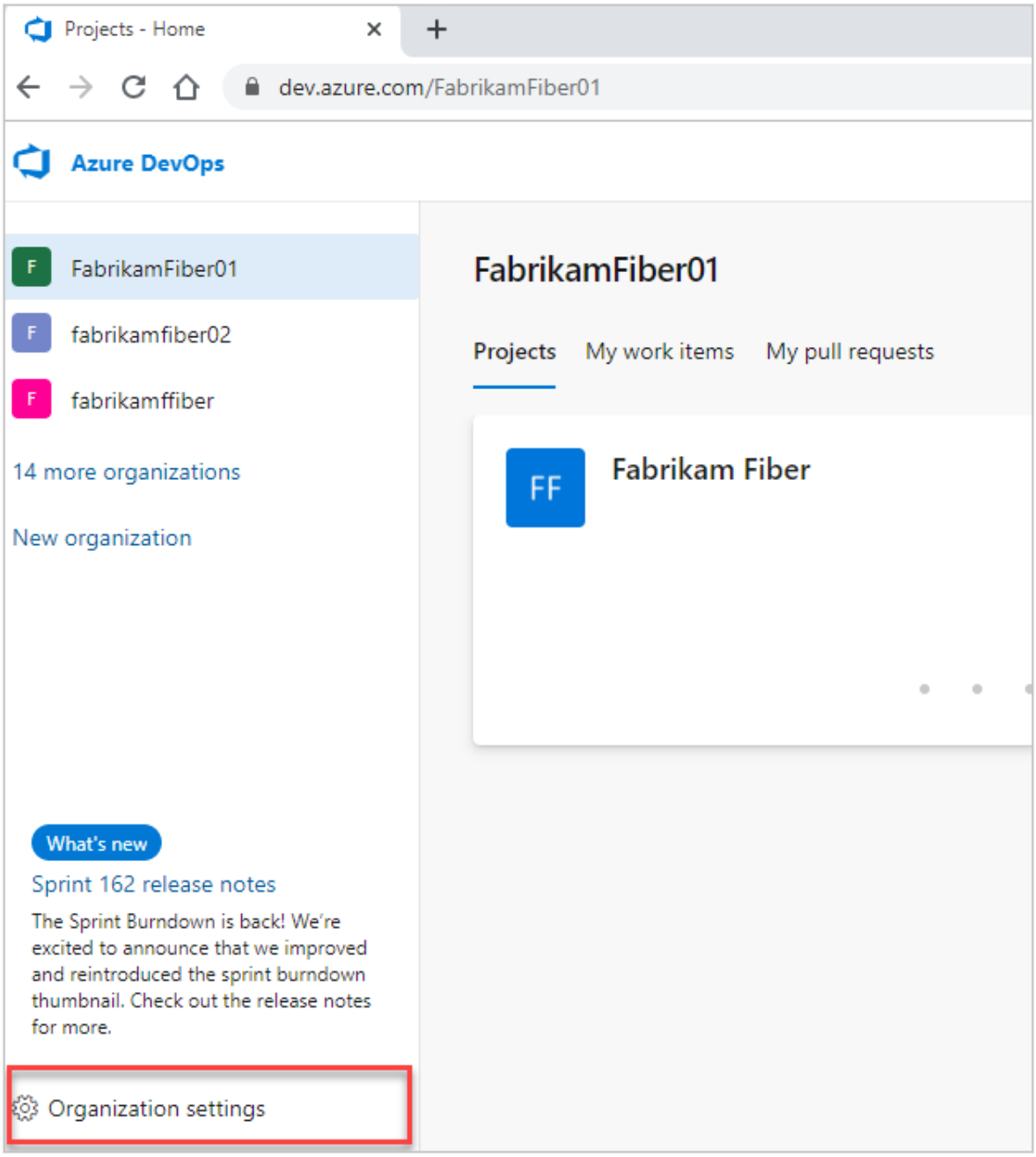
The general sequence of steps to support clone are:

1. Open **Settings>Process**
2. Choose the Hosted XML process you want to clone and select **Clone process to inheritance** from its context menu
3. Verify the customizations that are present in the inherited process created as part of the clone
4. Manually reapply customizations that were ignored during clone
5. Create a test project to verify customizations.

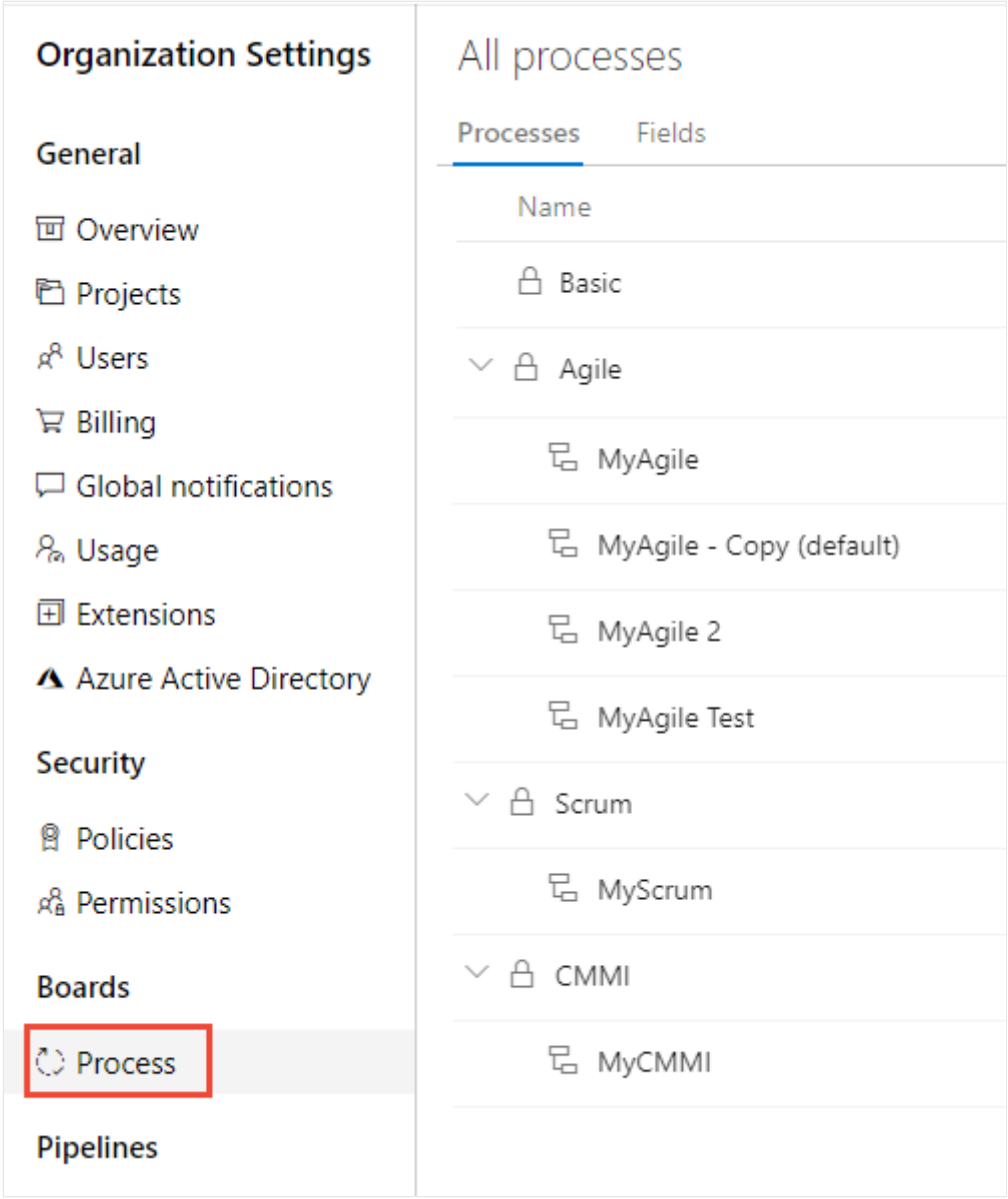
Open Settings>Process

You create, manage, and make customizations to processes from **Organization settings>Process**.

1. Choose the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.



2. Then, choose **Process**.

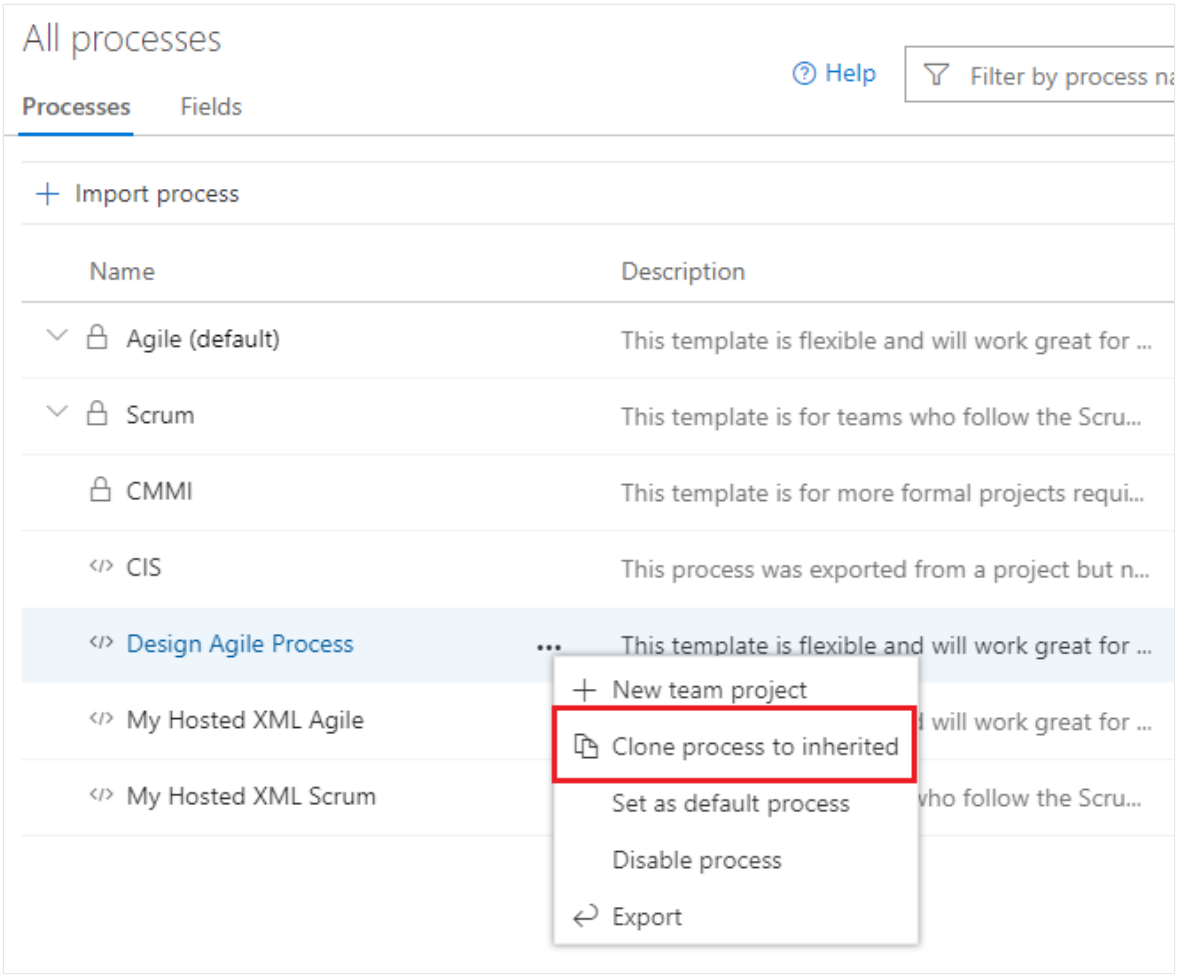


> **Important**
>
> If you don't see **Process**, then you're working from TFS-2018 or earlier version. The **Process** page isn't supported. You must use the features supported for the On-premises XML process model.

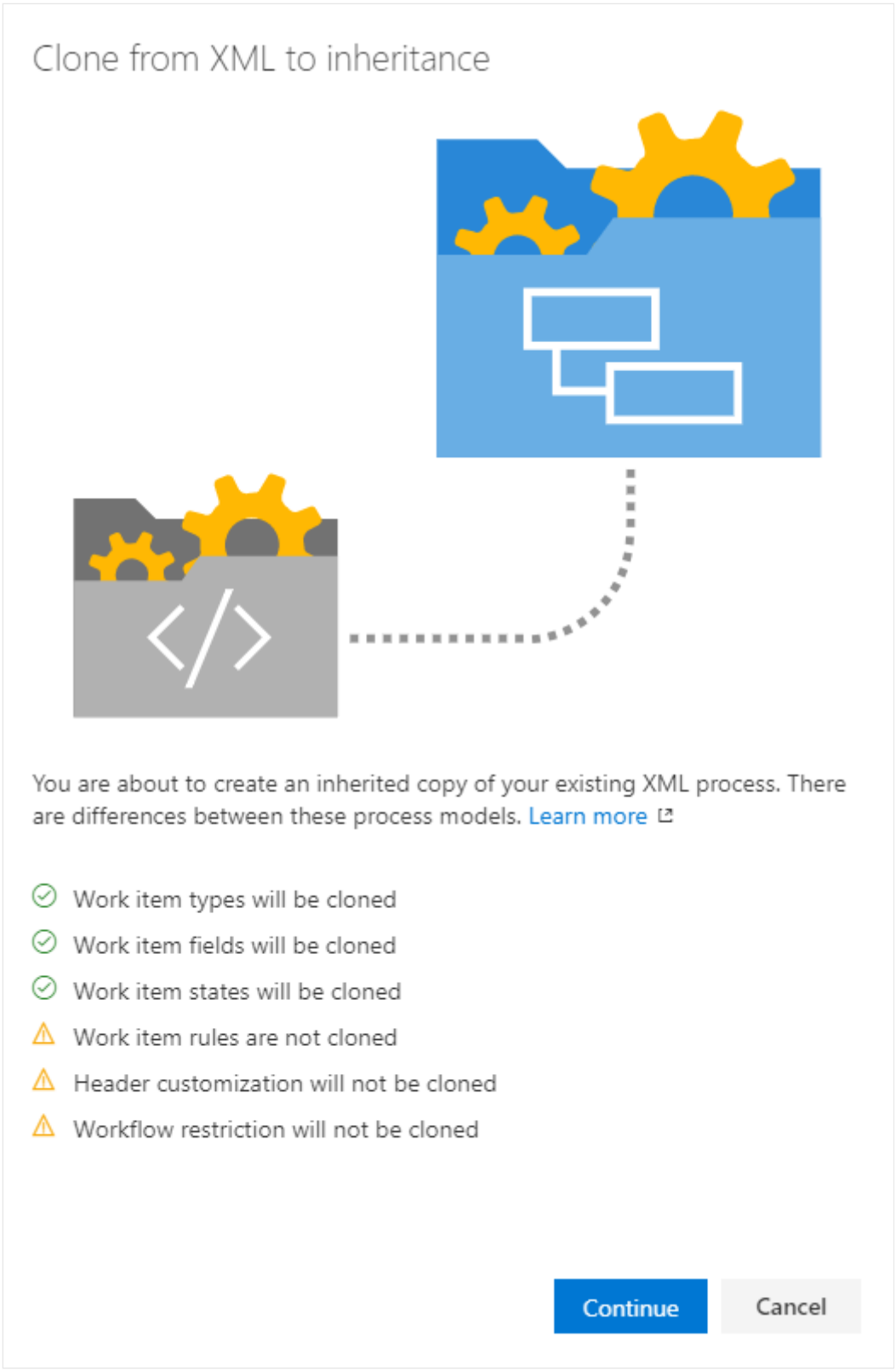
Clone a Hosted XML process to an Inheritance process

1. Open the ... context menu for the Hosted XML process and choose the **Clone process to inherited** option.

   Here we open the menu for the Design Agile Process.

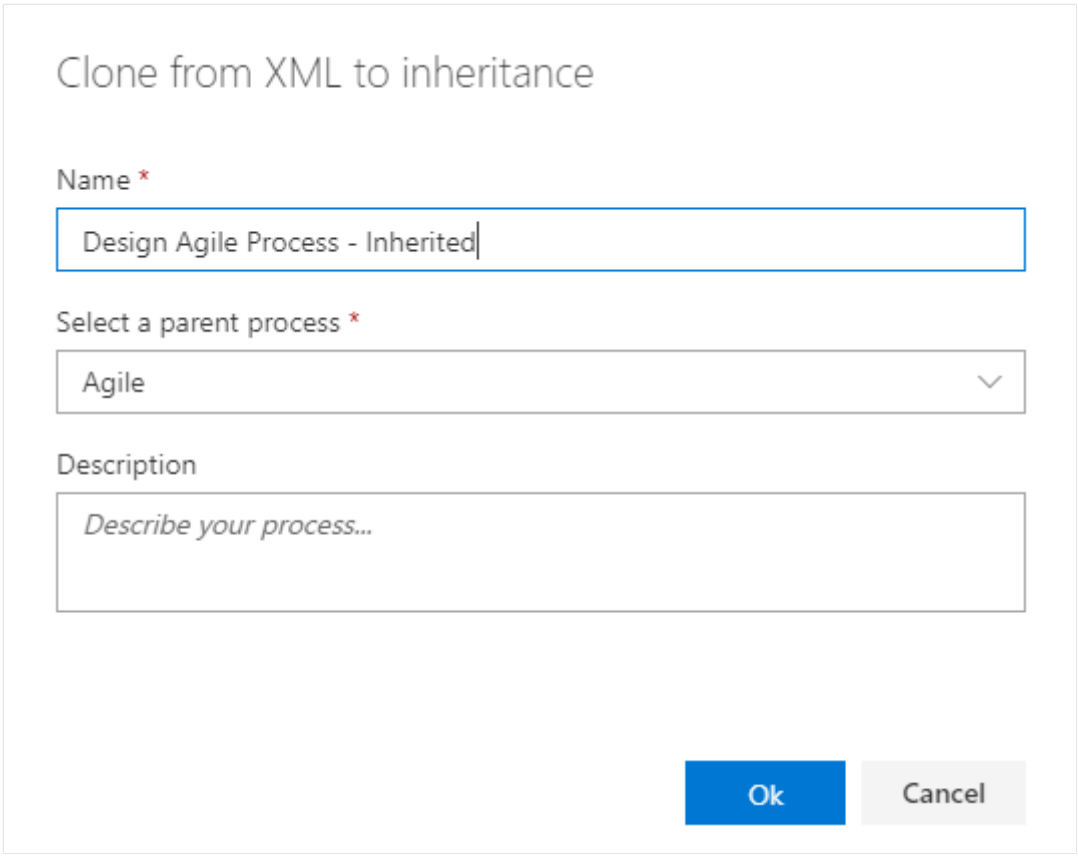


2. Review the information provided in the dialog that opens and then choose **Continue**.



3. In the next dialog, you can change the name of the inherited process to be created and optionally change the parent process and provide a description.

   Here, we accept the defaults provided by the system.



> **Important**
>
> The system performs a best effort to detect the correct parent process. Verify that the process selected meets your expectations, or change it as needed. Once you've completed the clone, you can't clone it a second time based on a different system process.

4. After you've confirmed that the information is correct, choose **Save**.

Post-upgrade customizations to make manually

The upgrade makes a best-effort attempt to reconcile the system process and the customizations made to the Hosted XML process. After you upgrade, we recommend you review the inherited process and reapply customizations manually.

- Create a test project: Use to verify the customizations preserved or reapplied to a process
- Update the default value for any field: define any default values you had previously defined
- Workflow states: verify the mapping of states to workflow state categories
- Custom rules: You can recreate select rules as needed. Rules for the Hosted XML process model don't map one-to-one to rules defined for an Inherited process.

Specifically:

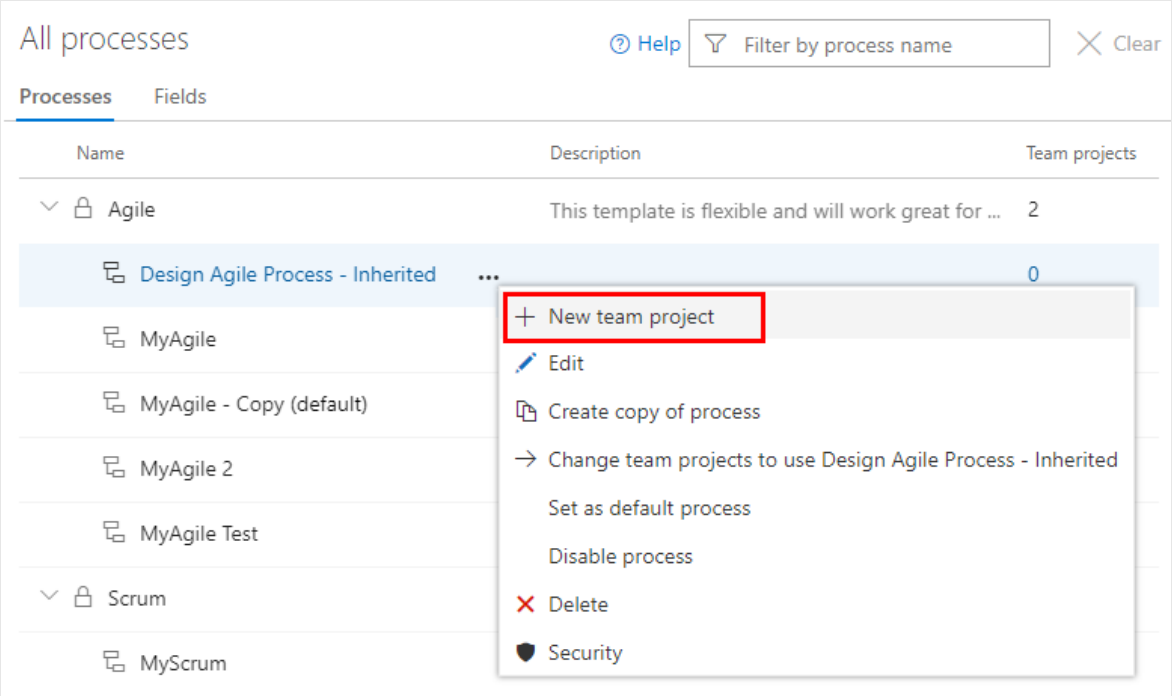
- Several rules are already defined in the system process or autogenerated. For example, certain system fields such as Changed By, Change Date, Closed By, Closed Date are governed by system rules.
- Some rules are now specified as field attributes, such as making a field a default or required.

- Disable work item types.
- Hide inherited fields or controls.
- Custom controls: verify that custom controls are applied as expected; disable or hide unwanted groups or page extensions.

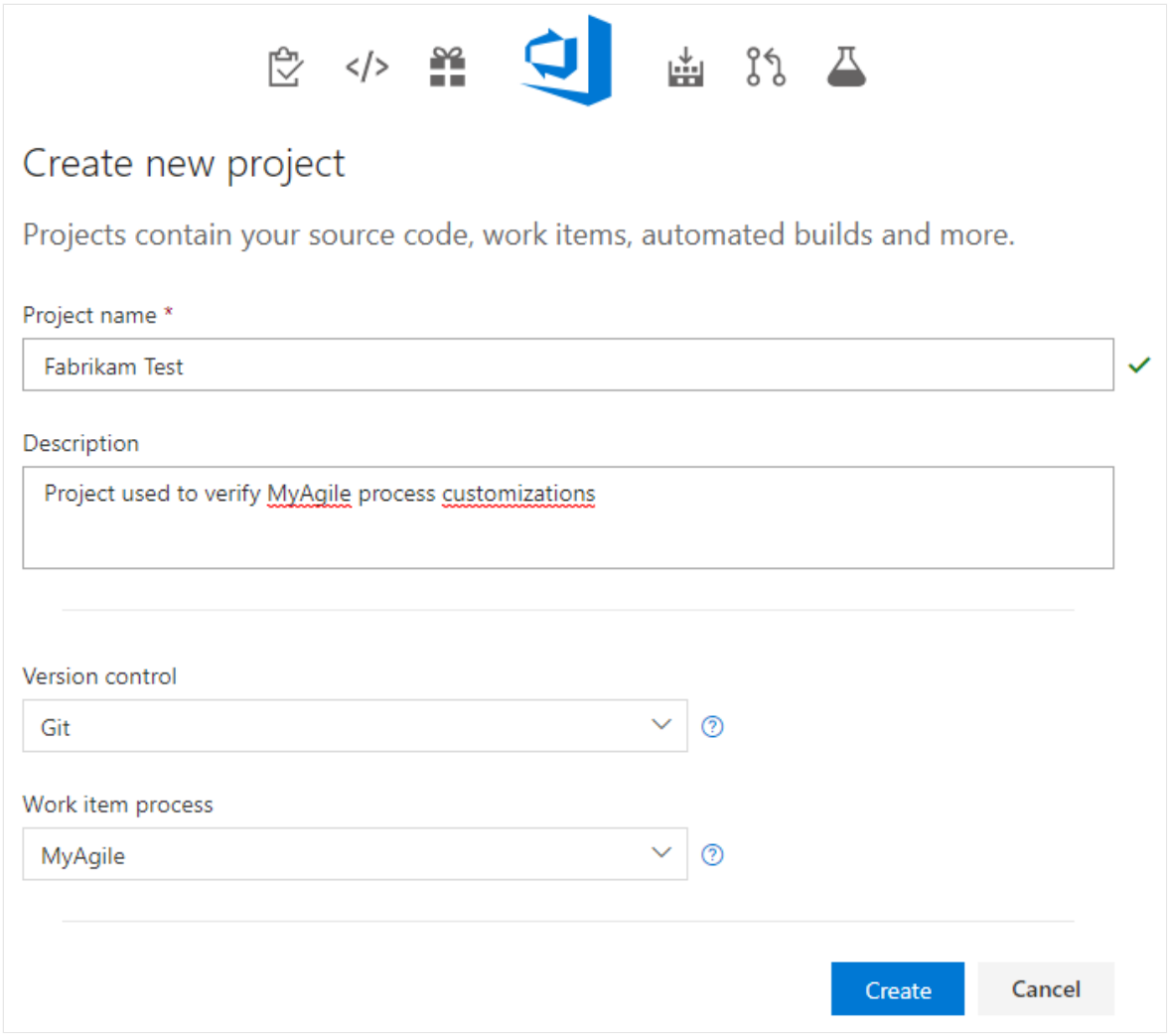
Verify the customization you made

We recommend that you create a test project based on the inherited process created in the previous step to verify the customizations preserved and any additional changes you make to it.

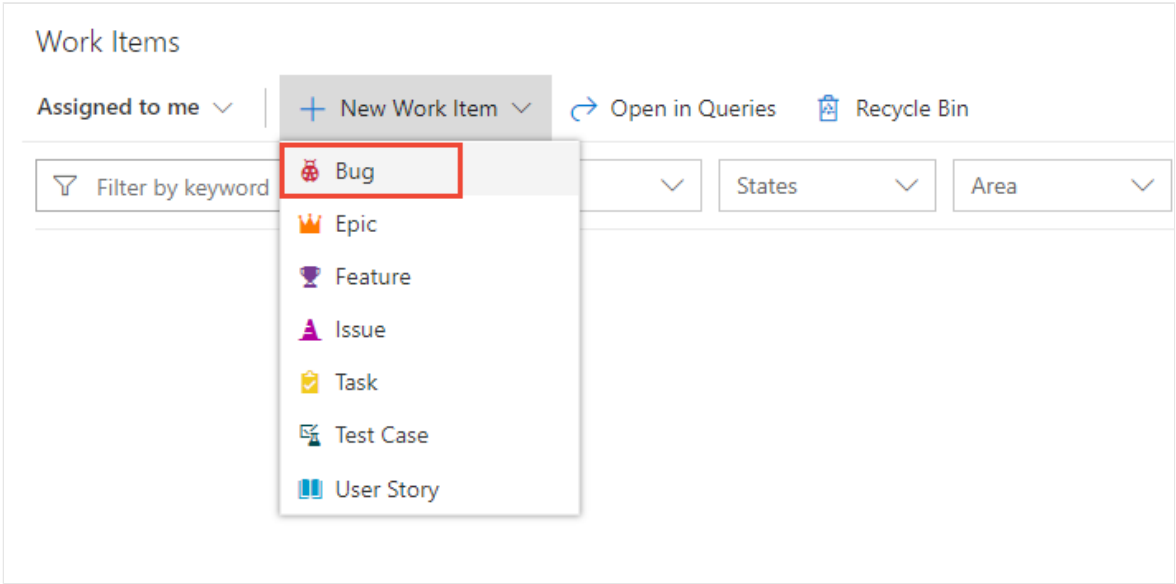
1. Open the **All processes** page, and choose the ... context menu for the process you want to use, and then select **New project**.



2. The Create new project page opens. Fill out the form.



3. Open the **Work>Work Items** page (user context) and choose **New Work Item** and select a WIT that you have previously customized. Here we choose **Bug**.



4. Verify that the customizations you expect appear on the form.

Try this next

Change a project from a Hosted XML process to Inheritance

FAQ

Q: Why are some of my fields missing on the form when I clone a Hosted XML process to Inherited?

During the cloning process we do our best to match your layout from Hosted XML and clone it into Inherited. However, if you have been creative with the form layout, we can't always replicate that same layout in Inherited. They system does a best attempt. Fields and layouts that are not understood get ignored. The fields however are still added to the process. You just need to manually add them to the form layout.

Change a project from Hosted XML to an inherited process

Article • 02/24/2023

Azure DevOps Services

Once you've cloned your Hosted XML process to an inherited process, you can change the project(s) from the Hosted XML process to the inherited process. You change a project from a Hosted XML process to its derived inherited process to start customizing the process through the user interface.

> **⊗ Caution**
>
> Choosing to clone a project from a Hosted XML process model to an inherited process is an irreversible operation. Before you change the process of an existing project from Hosted XML to the cloned inherited process, review **Supported operations when moving from Hosted XML process to inherited process** to understand which customizations are preserved and which are not before you change the process of a project from Hosted XML to an inherited process. Also, **create a test project** to verify the customizations preserved or reapplied to a process.

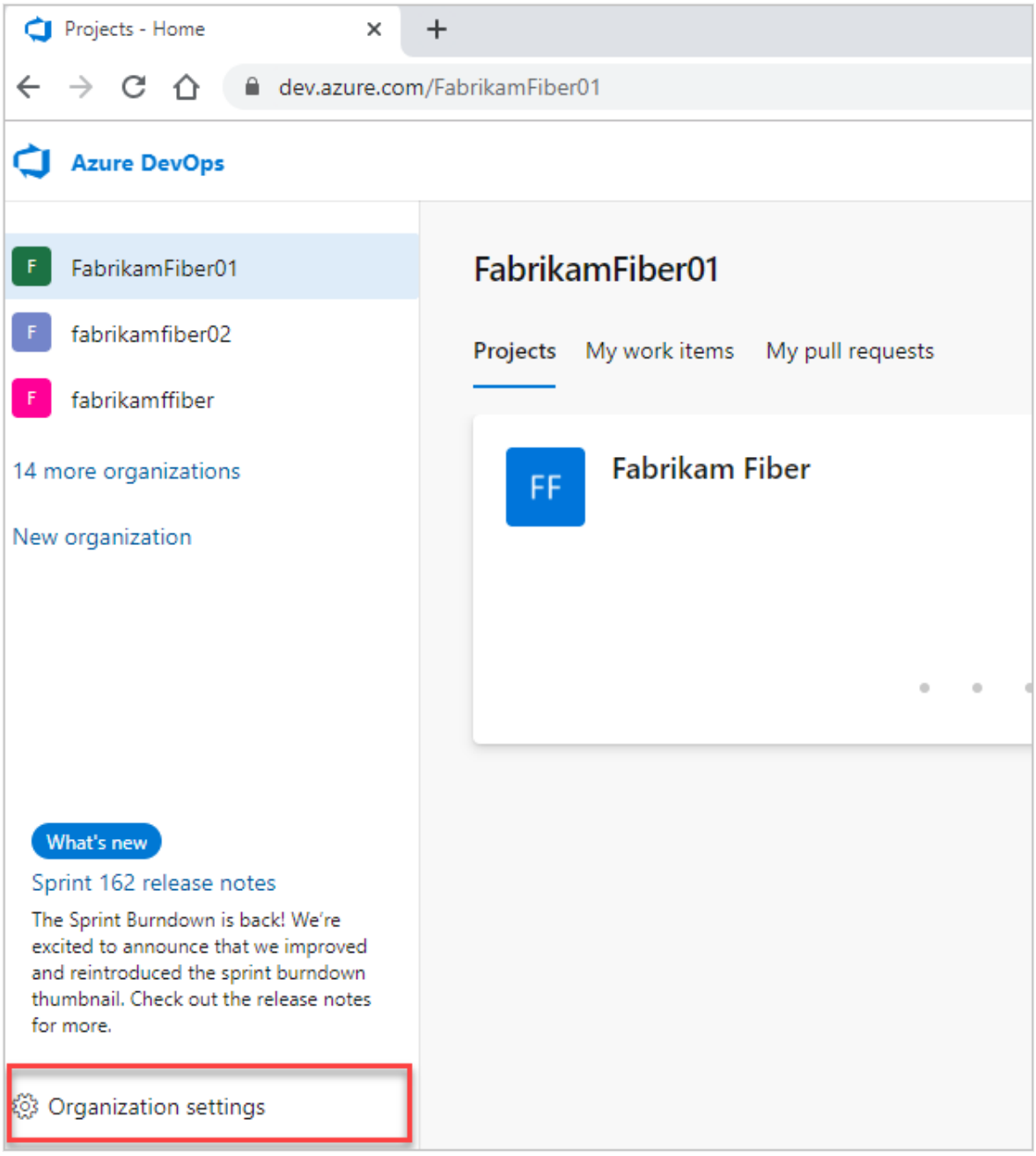
Prerequisites

- You must be a member of the Project Administrators group for the project you want to change, or a member of the Project Collection Administrators group.

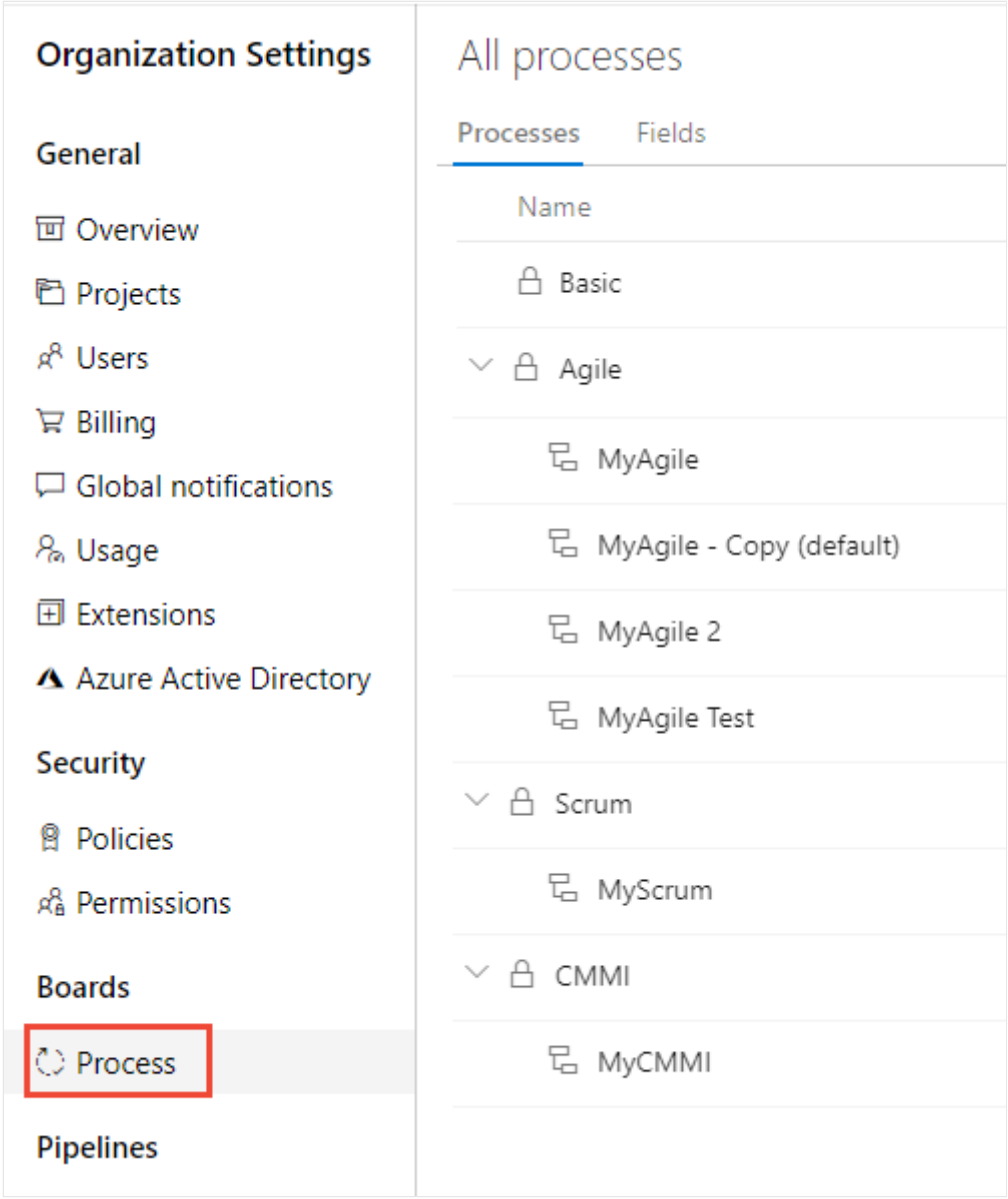
Open Settings>Process

You create, manage, and make customizations to processes from **Organization settings>Process**.

1. Choose the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.



2. Then, choose **Process**.

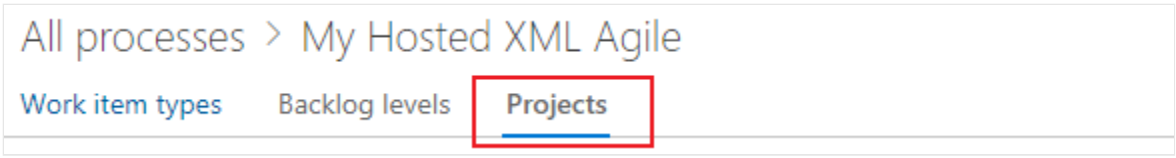


> **Important**
>
> If you don't see **Process**, then you're working from TFS-2018 or earlier version. The **Process** page isn't supported. You must use the features supported for the On-premises XML process model.

Change the project process to an inherited process

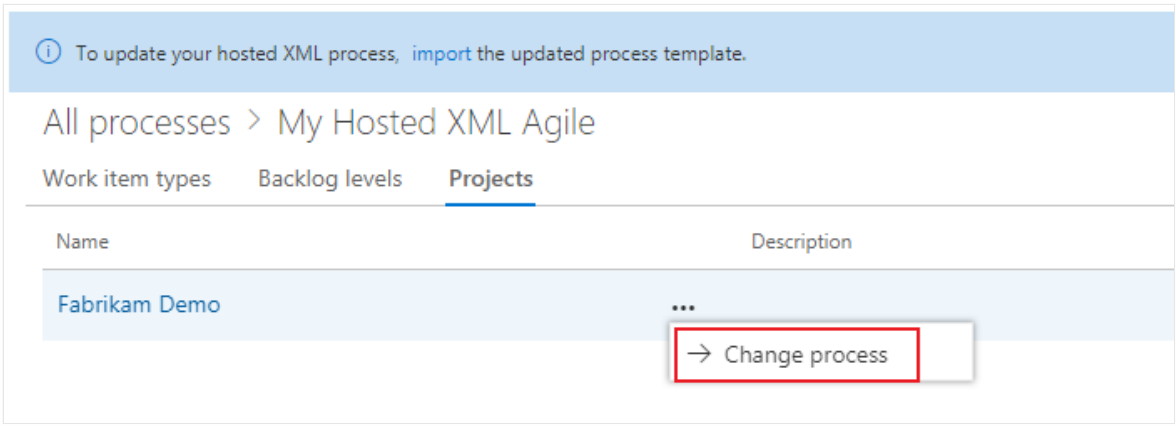
After you've verified your customizations, you can now apply the inherited process to your existing project.

1. From **Process**, choose the original Hosted XML process, then choose the **Projects** page.

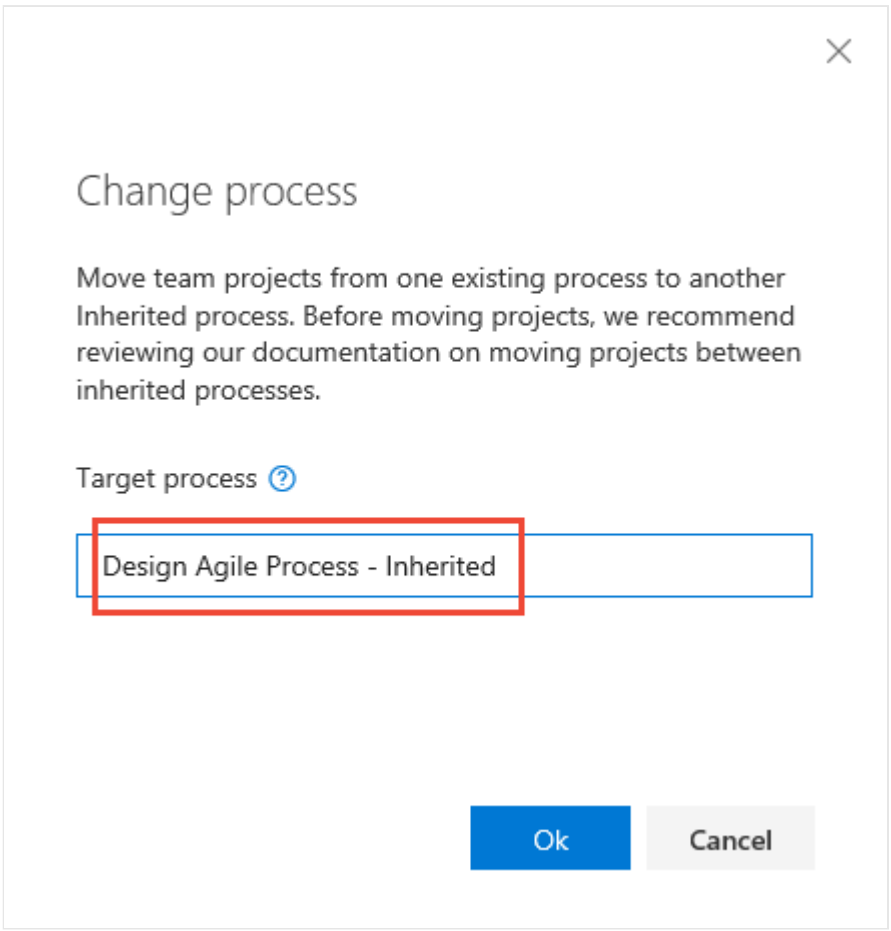


2. Open the ... context menu for the project and choose the **Change process**... option.

   Here we open the menu for the Fabrikam Test project.



3. Choose the inherited process that you created. The system lists only those processes that are valid for the selected project.



4. Choose **Ok**.

Related articles

- Supported operations when moving from Hosted XML process to an inherited process
- Clone a Hosted XML process to an Inheritance process

Process templates overview

Article • 05/02/2023

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

When you create a project, you choose a *process* or *process template* based on the *process model* for which your organization or collection was created. Before you choose a process for your project, understand the following terms:

Term	Description
Process model	Refers to the model used to support projects created for an organization or project collection. Only one process model is supported for a project at a time.
Process	Defines the building blocks of the work item tracking system and supports the *Inheritance* process model for Azure Boards. This model supports customization of projects through a What You See Is What You Get (WYSIWYG) user interface.
Process template	Defines the building blocks of the work item tracking system and other subsystems you access through Azure DevOps. Process templates are only used with the *Hosted XML* and *On-premises XML* process models. You can customize projects by modifying and importing process template XML definition files.

> ⓘ **Note**
>
> For more information, see the following articles:
>
> - **Configure and customize Azure Boards**
> - **Create a project using the process of your choice**
> - **Customize your work tracking experience (process models)**
> - **Manage inherited processes**

Process template customization is only supported for the Hosted XML and On-premises XML process models.

For more information, see About processes and process templates.

For on-premises deployments and Azure DevOps Services, you can upload/download process templates.

If you use the Hosted XML process model, you can import/export a process.

For details on creating a project using the process of your choice, see Create a project.

Related resources

- Customize a process template
- Process Template XML element index
- Define the work item tracking plug-in
- Control access to functional areas

Customize a process template

Article • 10/08/2024

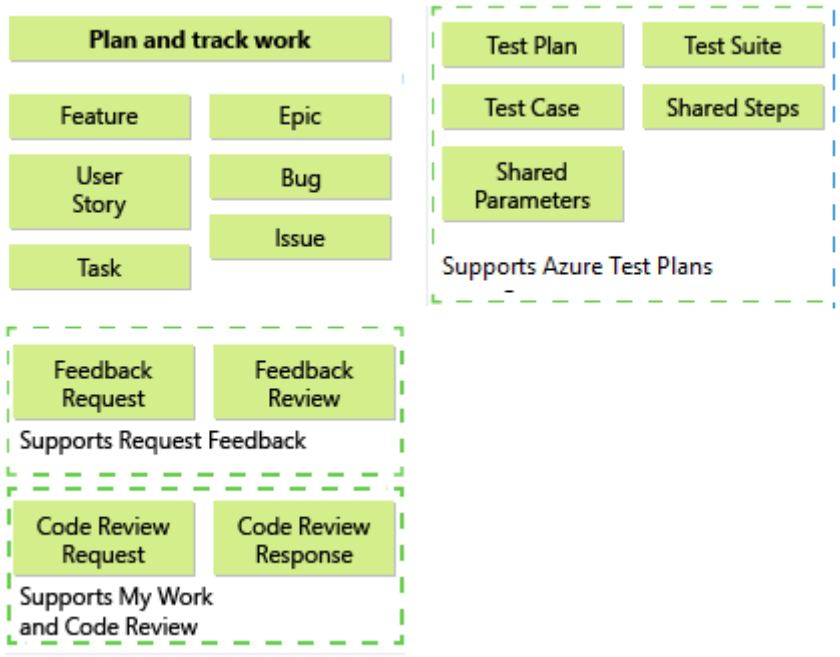
Azure DevOps Server 2022 - Azure DevOps Server 2019

Process templates define the objects and processes available to you when you create a project. By customizing a process template, you customize one of more objects. Common types of customizations you can make include:

- Add a new field to an existing work item type (WIT)
- Modify the pick list of values for a field
- Change the workflow states, reasons, transitions, and actions of a default or custom work item type
- Edit the layout of a work item form
- Add or remove a work item type
- Change the process configuration or defaults associated with Azure Boards hubs

> **Note**
>
> This article describes process templates used to create projects defined on Azure DevOps Services. If you're looking for **project templates** for software development, see Creating Visual Studio templates.

Default process templates define default configurations and work item types used by Azure Boards and Azure Test Plans. For example, the Agile process template defines the set of work item types shown in the following image.



Many of these artifacts depend on work item types that are used to track work. For example, data fields defined in the definition of Feature, Bug, User Story, or Task are also used to define work item queries. In addition to these artifacts, you can also define the initial project areas and iterations, security configuration, and other default settings that support test plans.

After you create a project, you can change configurations and customize artifacts. However, by customizing the process template before you create your projects, all resulting projects that you create from it matches a standard set of team processes. The main reasons you might want to customize a process template include:

- You plan to create several projects and you want to minimize repetitive tasks that you have to implement later in each project that you create
- You want to make sure all teams adhere to certain standards by providing the templates and structures within the toolset your software development teams use
- You need to update a custom process template to support using the Configure Features wizard after an upgrade

If you work with only a single project, then you might consider simply creating the project and customizing one or more objects later.

How are process templates used?

The primary use of process templates is to create a project. For the Hosted XML process model, it's also used to update a project. A project provides the set of objects, artifacts, and configurations defined in the interdependent set of template files. You use your project to organize source code, track work and information, build software, and support test activities.

Hosted XML process model

- Create a project
- Add or refresh a process

On-premises XML process model

- Create a project

Where should I start?

Before you start customizing a process template, get familiar with what you can configure and customize and then plan your changes accordingly.

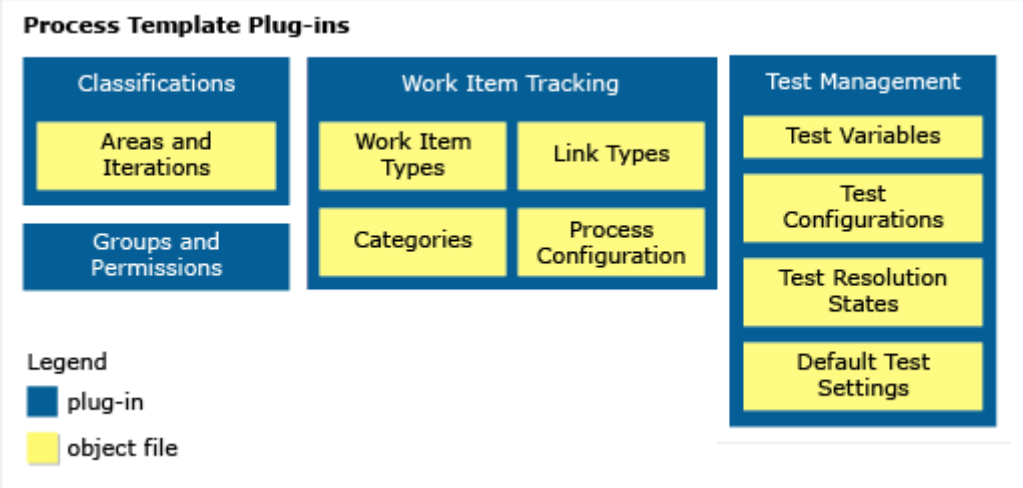
- If you're new to process templates, first review the default process templates.
- To get familiar with the file structure of a process template, review a description for each file or download a process template.

  You can modify the processes for your project after it's created. As you work with a project, the initial settings that the process template defined might no longer meet your needs.
- If you're most interested in customizing objects used to track work, which includes test plans, test suites, and test cases, review Customize your work tracking experience. The customizations you make by modifying an XML definition file for a project are the same types of customizations you make in a process template file.

  To add or modify types of work items, you can do so without changing the whole process template. You can make and test changes by using an existing project. For the On-premises XML process model, you can use the **witadmin exportwitd** and **importwitd** command-line tools to download and upload the XML definition files for work item types.
- If you're considering making extensive customizations, review how the changes you make impacts maintenance and upgrade of your projects.

Process template files and functional areas you can customize

Process templates consist of nine plug-ins. Each plug-in defines a set of tasks that execute and the screens that appear when you create a project. Tasks set permissions, create folders, upload files, activate sites, or set other configurable variables. Plug-ins also specify the dependencies that a task has on the successful completion of other tasks.



To customize a process template, you customize one or more files associated with a functional area. While customizing any one object is fairly simple, make sure that you don't break any inter-dependencies when you customize. The ProcessTemplate.xml plug-in file defines which plug-ins to include in the template. This file contains all the task groups that you want to run to create a project. Each task group references a subordinate XML plug-in file where the specific tasks for that plug-in are defined.

- **Teams, groups, permissions, & area and iteration paths**
  - Area and iteration paths
  - Groups, teams, members, and permissions
- **Work item tracking**
  - WIT definitions
  - Categories
  - Process configuration
  - Link types
- **Test management**
  - Test environments, configurations, resolution states, and settings

Plug-in dependencies

Many objects rely on the definition of other objects within a process template. For an overview of required plug-ins and plug-in dependencies, see Define dependencies for task groups and tasks.

Plug-in and naming restrictions

When you add objects to a process template, make sure that you label them correctly so that you avoid XML validation errors.

- Restrictions are put on the names or labels of most Team Foundation objects. For an overview of naming restrictions that apply to process templates, security groups, area and iteration nodes, work item types, and work item fields, see Naming restrictions.
- Most process template components that you customize affect only the project that you create by using the process template. The exceptions to this rule are global lists, link types, and work item fields. These objects are defined for a project collection.
- Each work item field has an associated field reference name that uniquely identifies each field. The reference name can't be changed after it's assigned.

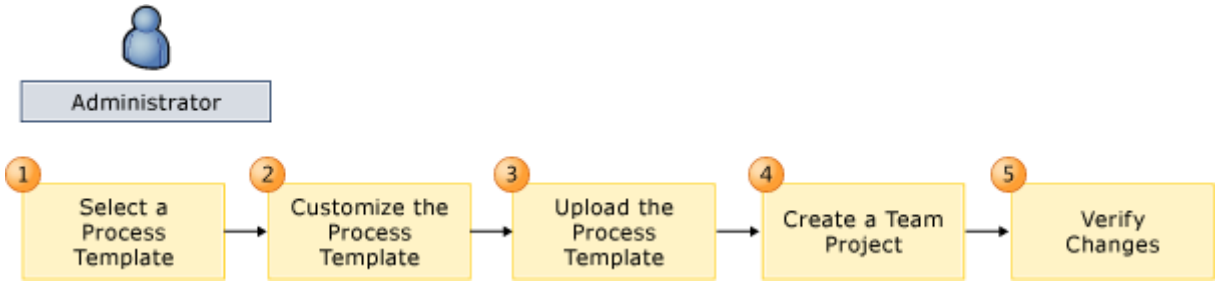
Work item field names, link type names, and global lists are scoped to a project collection. If you customize any of these objects, the change is reflected in all projects that are defined in the collection and in the work item types that contain that work item field.

- The maximum size of a process template is two gigabytes. When you customize a process template, make sure that your changes don't increase its size beyond that value.

Steps to customize a process template

Customizing a process template is an iterative process. You need a project collection that is defined on a server that is running Azure DevOps Server where you can test your process template to make sure that it was customized correctly.

To customize a process template, you first download an existing process template, modify or add files, upload the process template files, and then verify your changes.



Expand table

Step	Task
1	Download a process template. Before you can customize a process template, you must download it to your local computer. To minimize the modifications that you must make, select a template that most closely matches your team processes. In general, you choose a process template based on types of work items and workflow.
2	**Modify or add files**. You customize a process template by modifying, deleting, or adding files that are defined for a process template. You customize a plug-in or definition file by modifying its XML content. Each plug-in file and type definition file must conform to its XML schema definition. The first time that you customize a process template, make a small change. If you make many changes without a good understanding of how the changes might affect your template, you risk encountering multiple mistakes difficult to debug. Make sure that the name of your process template is unique. If you download a process

Step	Task
	template, make changes, and upload it, you must change its name or it over-writes the existing process template from the project collection.
3	Upload a process template. After you customize your template, upload it to the project collection where you create the project. Ideally, you should use a project collection that isn't used by other projects. By working in a test-bed project collection, you avoid introducing a change that might collide with existing team processes that are still under development. Also, the project collection should support the same resources that you want to access, such as a project portal and a reporting site. Make sure that the name of your process template is unique. If you downloaded a process template from a project collection, made a change, and are now uploading the template, you must change its name or delete the existing process template from the project collection. The upload process performs a verification check to make sure that the XML is valid. If you receive any errors when you try to upload the process template, the changes that you made caused the error. Review your changes, and correct any XML syntax errors that you find.
4	Create a project. To test new process templates, you must create a project. If any errors occur, view the log for project creation. It contains a list of the tasks that it tried to run and shows which tasks failed. You can map failed tasks back to the XML to determine the cause of the errors. You can clean up unnecessary projects by using the TFSDeleteProject command-line tool.
5	**Verify changes to process templates**. Before you put your process template in production mode and use it as the basis for several projects, you should verify that it's well-defined. You perform this task by systematically verifying that each object and artifact works as expected.

Tools that support customizing a process template

You can customize a process template using one of the following tools:

- Any text editor or XML editor to modify XML files.
- The Process Editor Tool.
  Based on the version of Visual Studio you have installed, get the Process Editor Tool from one of the following extensions.
  - **Visual Studio 2019 & 2022:** As of now, there's no specific Process Template Editor extension available. But, you can still customize process templates using

other tools such as any text editor or XML editor. You can also use the Azure DevOps web portal for certain customizations, especially for inherited processes.

- **Visual Studio 2017**: TFS Process Template Editor. You can also use this version of the Process Editor to modify the old-style work item forms. You can't use it to edit forms associated with the new web forms.
- **Visual Studio 2015**: TFS Power Tools.

Related articles

- Learn about process templates
- Explore process template and plug-in files
- Configure features after upgrading
- Review process template plug-ins and XML elements
- Reference XML elements

ⓘ **Note:** The author created this article with assistance from AI. Learn more

Feedback

Was this page helpful? Yes No

Provide product feedback

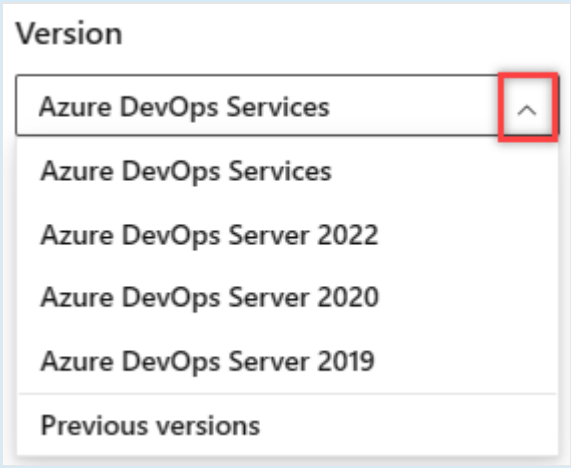
Upload or download a process template

Article • 03/25/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

When you create a project, a process is used to specify the work item types to configure and other settings. Each process template primarily defines the building blocks of the work item tracking system for Azure Boards. For the Hosted XML and On-premises XML process models, you can upload a process template to either update an existing project or create a project. For the Inheritance process model, you customize work tracking through the web portal. To learn more about these different processes, see Customize your work tracking experience.

Important



Select the version of this article that corresponds to your platform and version. The version selector is above the table of contents. Look up your Azure DevOps platform and version.

Prerequisites

To manage process templates, you must be a member of the **Project Collection Administrators** group, To get added to this group, see Change project collection-level permissions.

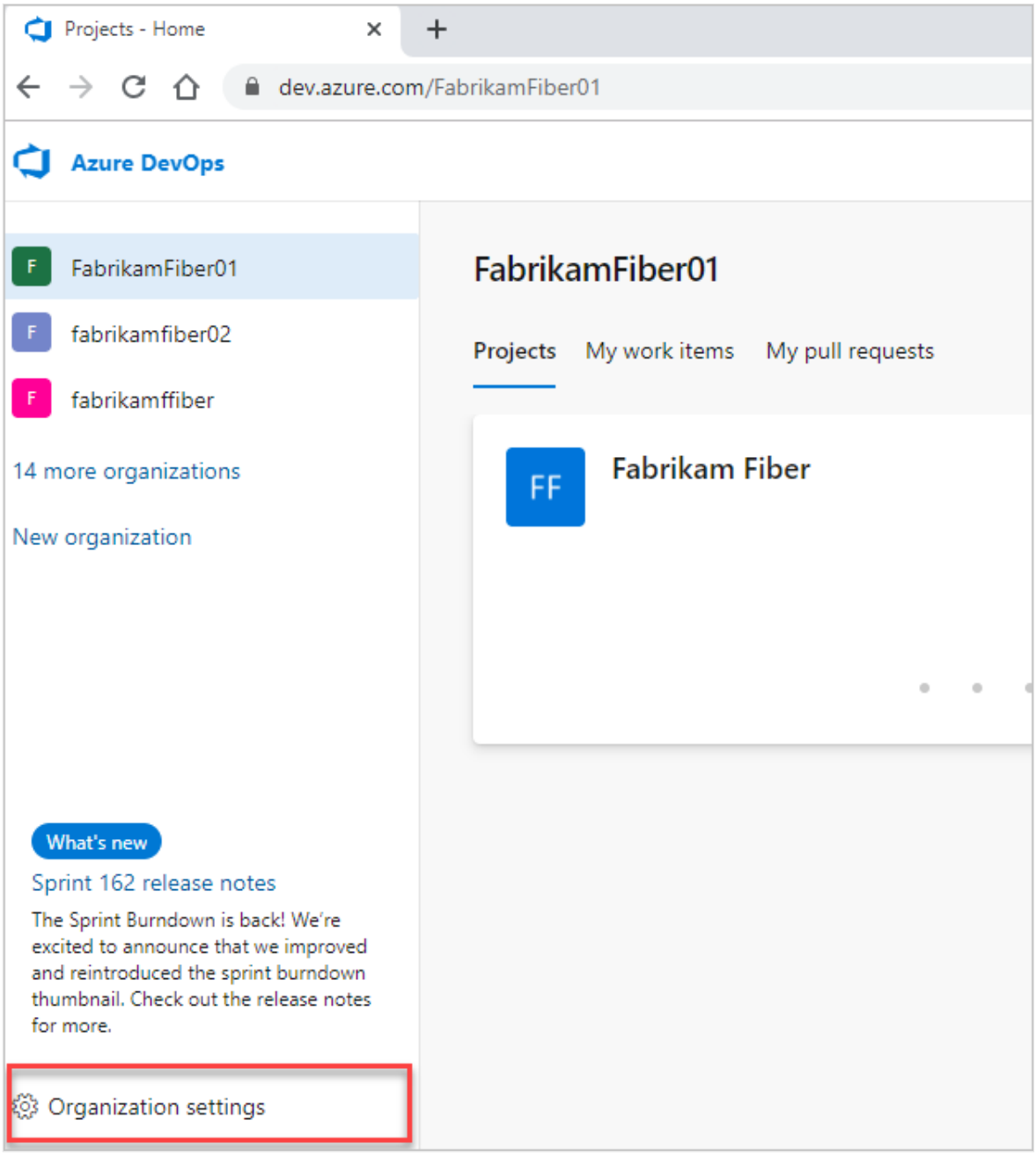
Important

Uploading and downloading Inherited processes isn't supported. To manage Inherited processes, see About process customization and inherited processes.

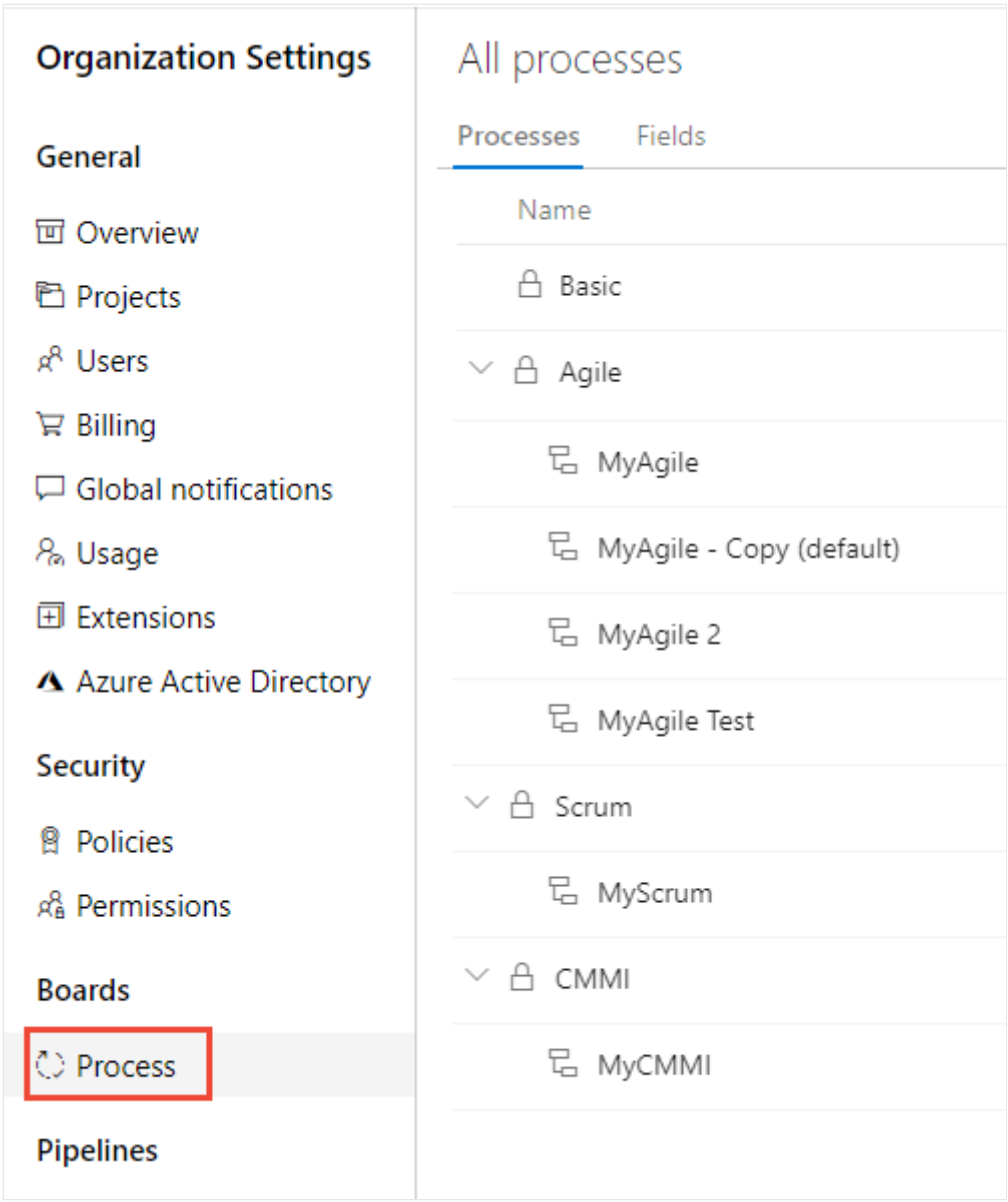
Open Settings>Process

You create, manage, and make customizations to processes from **Organization settings>Process**.

1. Choose the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.



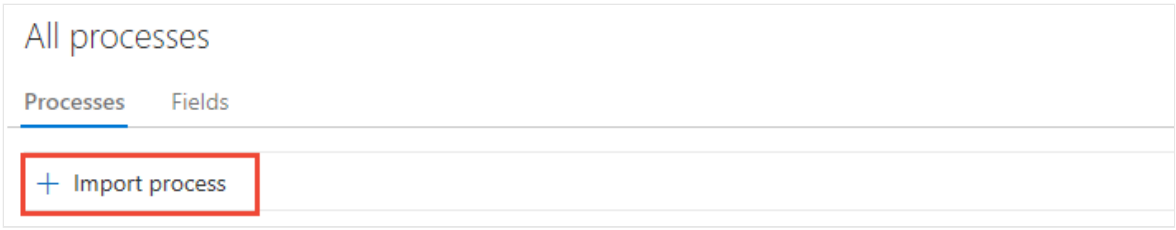
2. Then, choose **Process**.



> **Important**
>
> If you don't see **Process**, then you're working from TFS-2018 or earlier version. The **Process** page isn't supported. You must use the features supported for the On-premises XML process model.

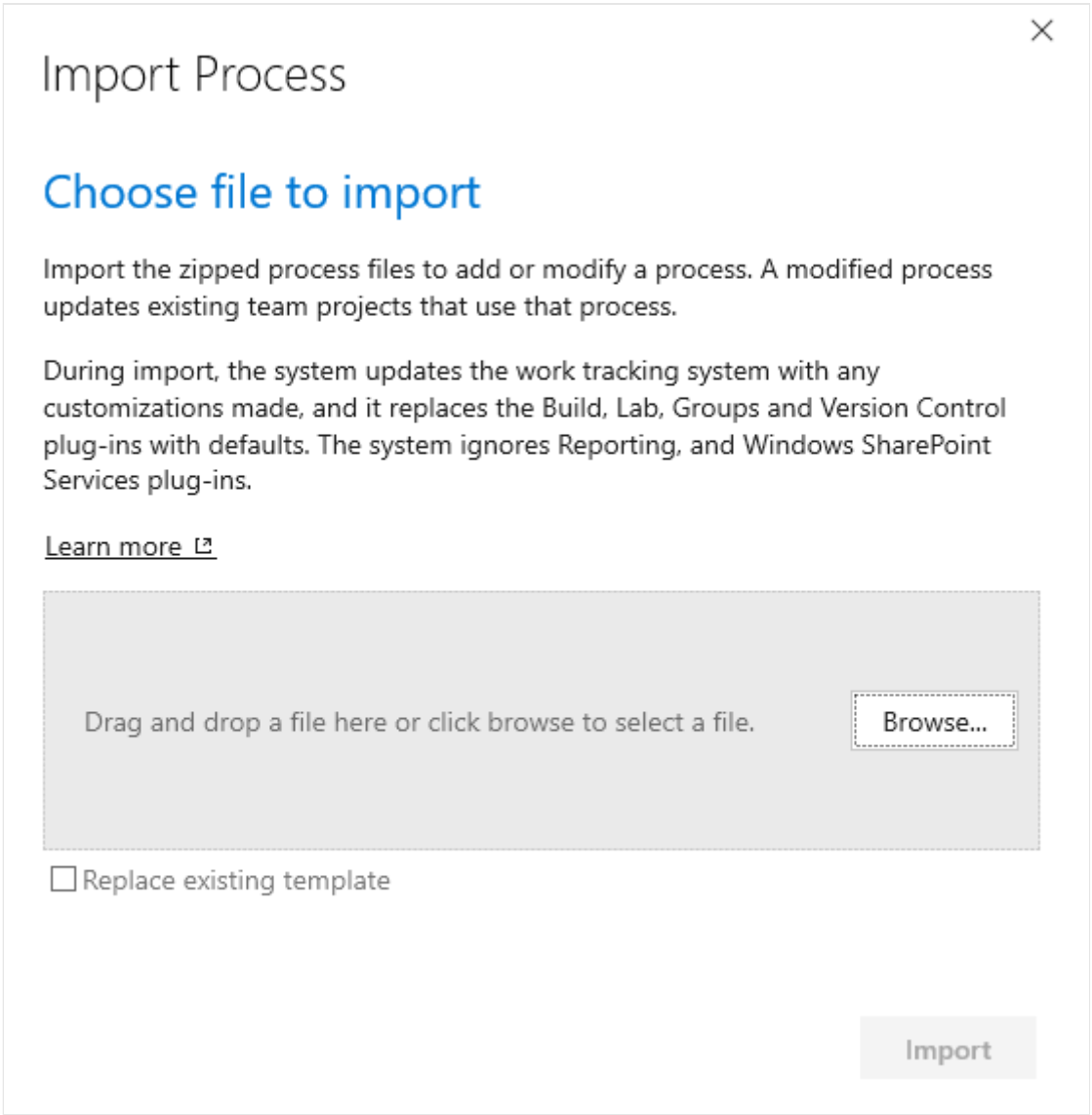
Import a process template (Hosted XML process)

1. From the **Processes** tab, choose **Import process** and then drag-and-drop or browse to the zip file of the process you've customized.



> **Note**
>
> If you don't see the **Import process** link, then your organization isn't set up to support the Hosted XML process model. You should work with the Inheritance process model for your customization needs. The Hosted XML process model is only supported if your account was created through the Data Import Service.

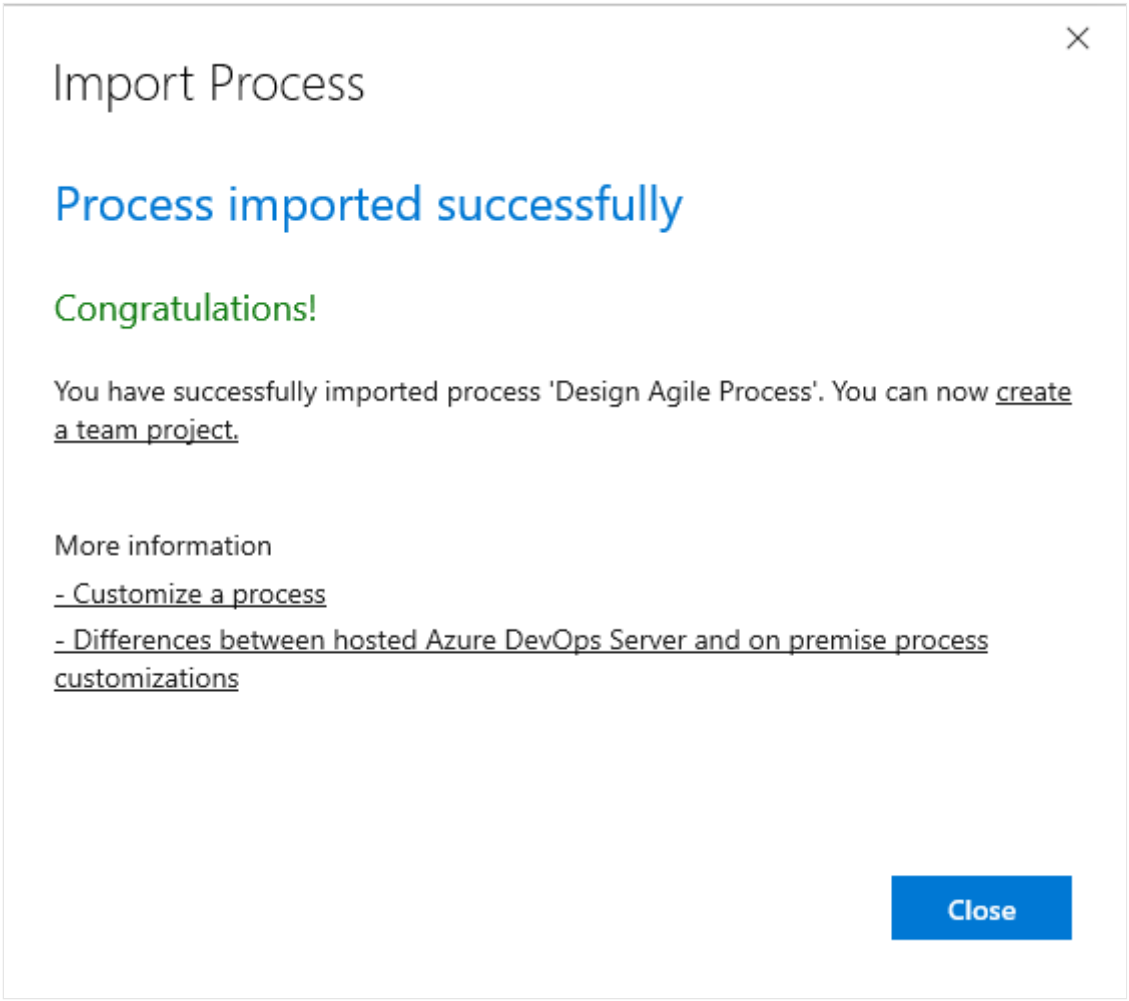
2. Choose the file to upload. Your custom process must meet specific constraints to pass validation checks during import.



Check the **Replace existing template** if you're updating an existing template. The import process overwrites any template that has the same name as the one you import and requires that you check this box to confirm you want it replaced.

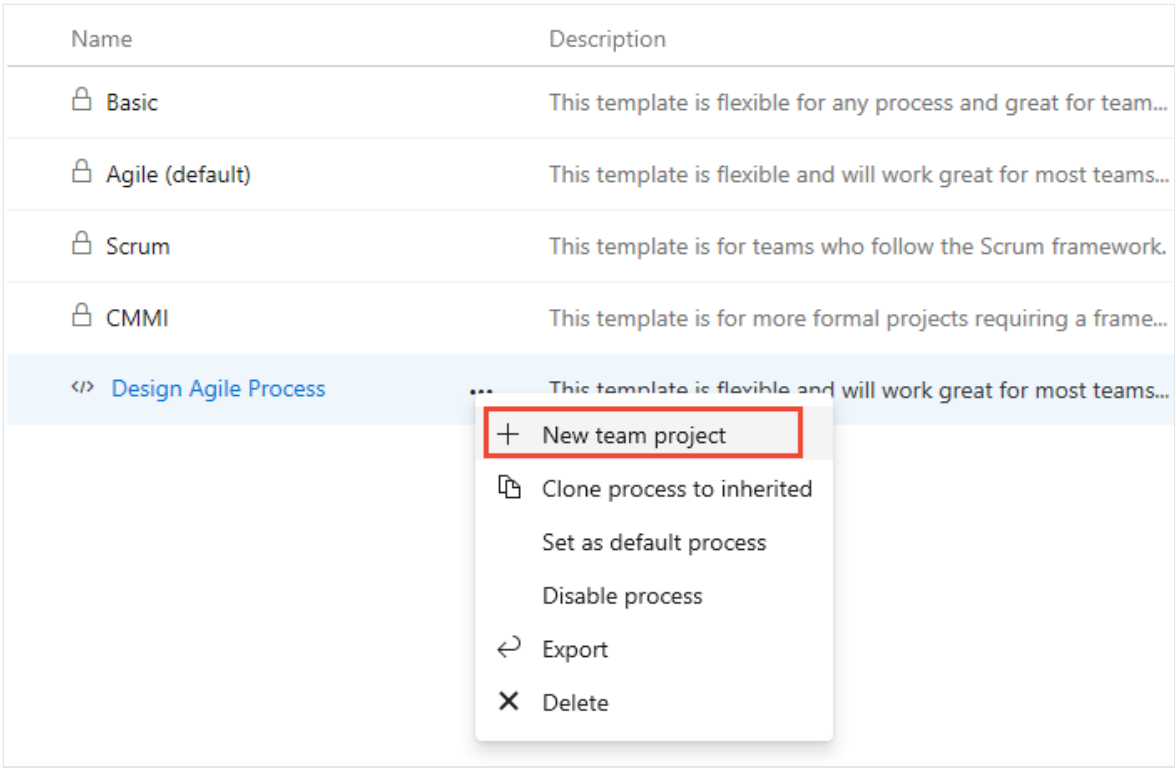
> **Important**
>
> You can't update one of the locked processes: Agile, Basic, CMMI, and Scrum.

3. Upon successful import, you'll see the following message.

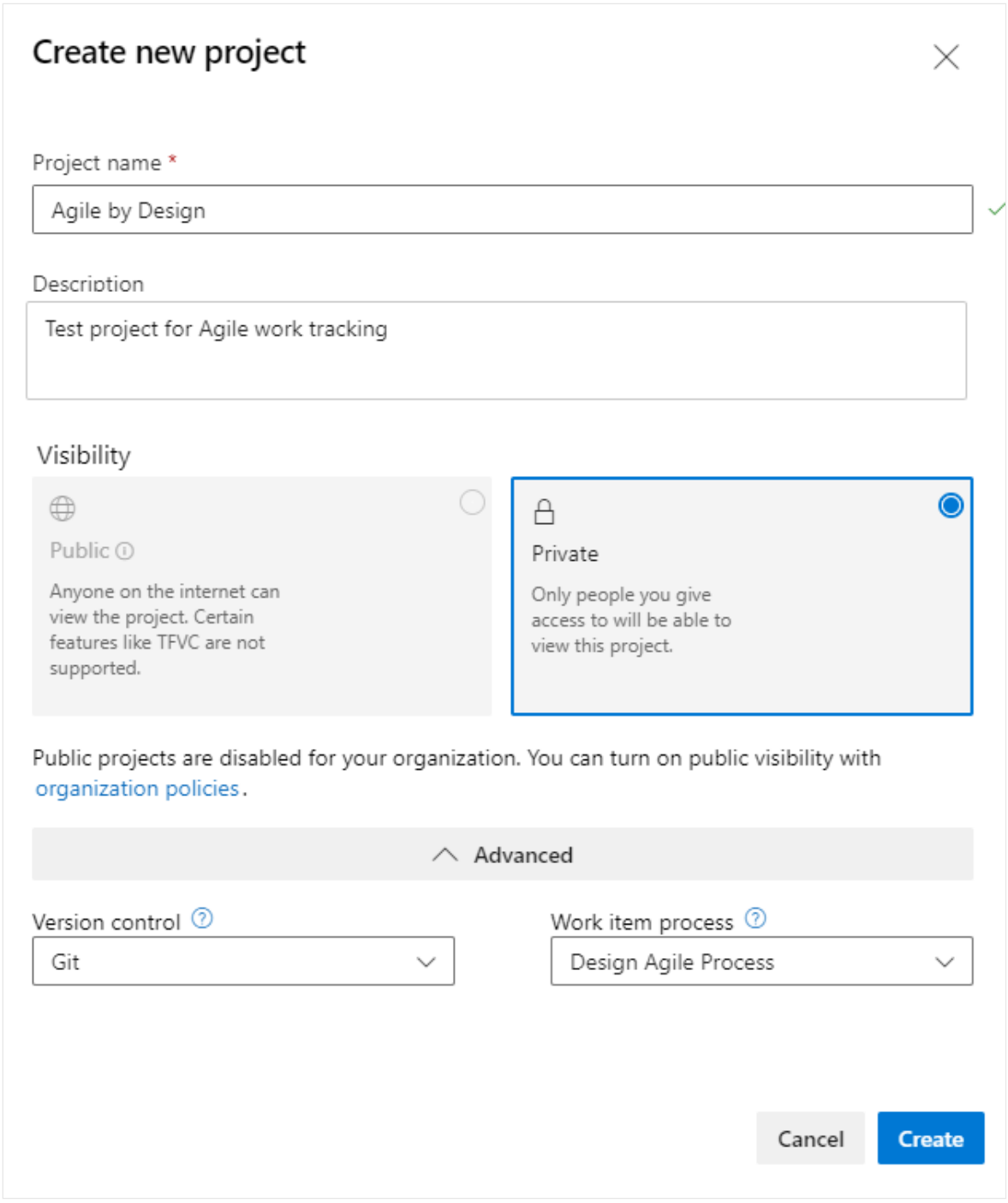


If the process doesn't pass the validation checks performed on import, you'll receive a list of error messages.
Correct each error and then retry the import.

4. You can immediately create a project using the newly imported process.



5. Complete the form that appears. To learn more about the different options, see Create a project.



For other Hosted XML process management tasks, see Import and export a Hosted XML process.

Correlate a process template with an existing project

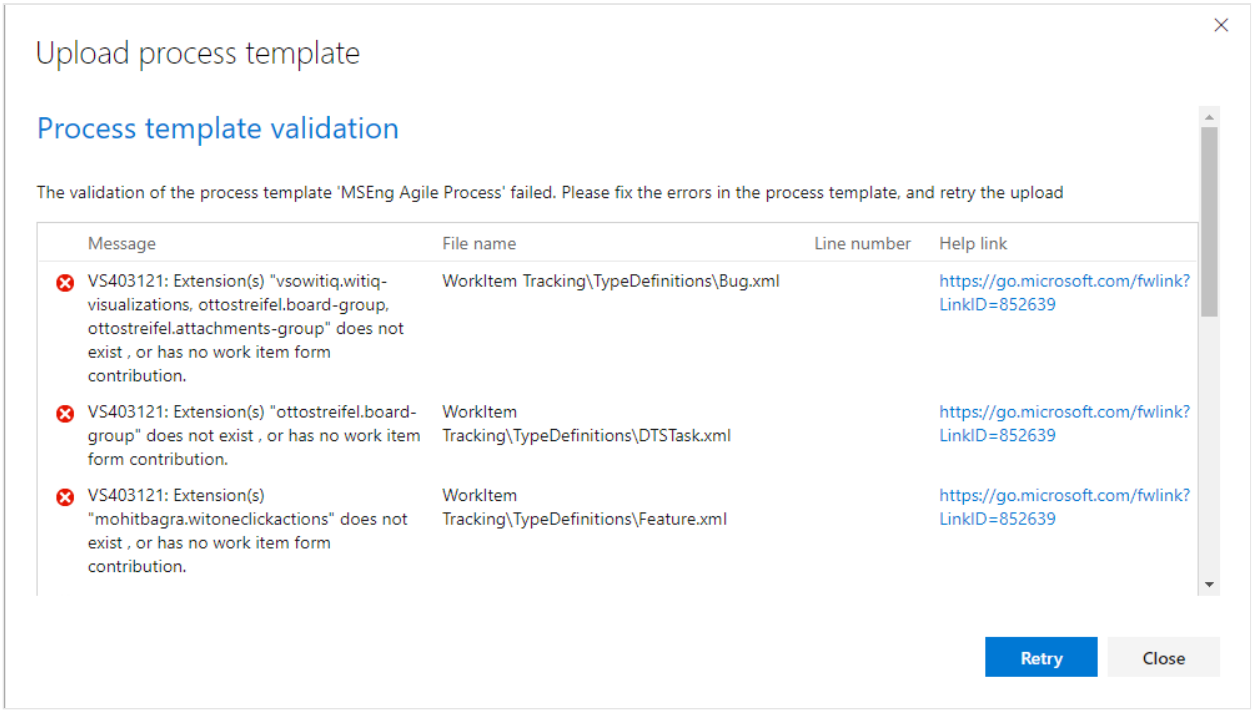
To determine the type of process template that was used to create your project, review the work item types that appear in the **New Work Item** menu for Team Explorer and then compare them with the work item types in the following chart. If your work item types differ from those items shown, then a custom process template might have been used.

Expand table

Agile	Basic	CMMI	Scrum
Bug	Issue	Bug	Bug
Task	Task	Task	Task
User Story	Issue	Requirement	Product Backlog Item
Epic	Epic	Change Request	Epic
Feature		Epic	Feature
Issue		Feature	Impediment
Test Case		Issue	Test Case
		Review	
		Risk	
		Test Case	

Process template restrictions and validation checks

If you upload the process template through the web portal, you may come across errors such as those shown in the following image.



Resolve each error and then retry the upload process. You may find additional information about resolving process template errors from Resolve validation errors for process import.

The following validation checks must pass when uploading a process template.

- Process template names must be unique and 256 Unicode characters or less. Same-named templates overwrite existing templates. Also, names can't contain the following characters: . , ; ' ` : / \ * | ? " & % $! + = ( ) [ ] { } < >. For more restrictions, see Naming restrictions.
- Process template folders can't contain any .exe files. If they do, the process template may upload successfully, however project creation fails.
- Process template total size should be 2 GB or less, or project creation fails.
- The upload process runs a partial verification check to make sure that the XML of each process template XML file is valid. If you receive any errors when you try to upload the process template, review the XML to determine the cause of the error. Duplicate tags in an XML file can cause errors. If there's no error in the XML, check to make sure all the appropriate files are included in your process template in the correct folder locations.
- If XML definition files reference an extension, then the extension must be installed and enabled in the collection.

> **Important**
>
> The schema definition for process templates uses a mix of camel-case and all capitalized elements. If you encounter errors when validating your type definition files, check the case structure of your elements. Also, the case structure of opening and closing tags must match according to the rules for XML syntax. See also **Process template plug-ins: Index to XML element definitions**.

Related articles

Periodically, updates are made to the process templates to support new features. For more information, see Release Notes for Azure DevOps Server.

- Create a project
- Customize a process template
- Process template and plug-in files
- Import and update a process (Hosted XML)
- Customize your work tracking experience

Feedback

Was this page helpful? Yes No

Provide product feedback

witAdmin: Customize and manage objects for tracking work

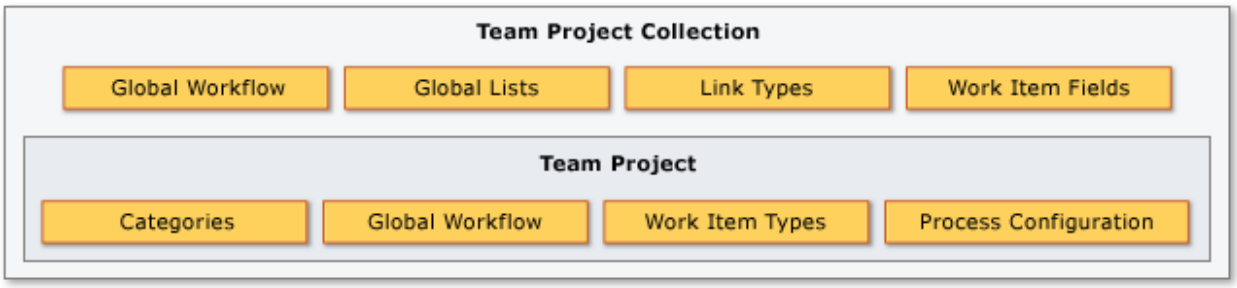
Article • 03/25/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

> **ⓘ Important**
>
> For the On-premises XML process model, you can use **witadmin** to list, import, export, and modify a process for a project. For the Inherited and Hosted XML process models, you can only use **witadmin** to list and export process information. For an overview of process models and what is supported, see Customize your work tracking experience.

You can change how you track your team's progress by creating and customizing objects that track work items. By using the **witadmin** command-line tool, you can create, delete, import, and export objects such as categories, global lists, global workflow, types of links, and types of work items. You can also delete, list, or change the attributes of work item fields.

As the following illustration shows, you manage categories and types of and work items for each project. You manage global lists, types of links, and fields in work item for each project collection. You can customize global workflow for a project or a collection.



For most commands, you must be a member of the Project Administrators or Project Collection Administrators groups. For more information, see Change project collection-level permissions.

Run the `witadmin` command-line tool

To run the `witadmin` command-line tool, open a Command Prompt window where Visual Studio is installed. The `witadmin` command-line tool installs with any version of

Visual Studio. You can access this tool by installing the free version of Visual Studio Community or Visual Studio Team Explorer.

> **Note**
>
> To connect to Azure DevOps Services, we recommend that you use the latest version of Visual Studio or Visual Studio Community.

For Visual Studio 2022

`%programfiles(x86)%\Microsoft Visual Studio\2022\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `Professional` or `Enterprise` in place of `Community`, depending on the version you installed.

For Visual Studio 2019

`%programfiles(x86)%\Microsoft Visual Studio\2019\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `Professional` or `Enterprise` in place of `Community`, depending on the version you installed.

For Visual Studio 2017

`%programfiles(x86)%\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `TeamExplorer` or `Professional` in place of `Enterprise`, depending on the version you installed.

On a 32-bit edition of Windows, replace **%programfiles(x86)%** with **%programfiles%**. You can install Visual Studio Community (which provides access to Team Explorer) or Visual Studio Team Explorer 2017 for free.

Global parameters

You can use the following parameters to display help text for `witadmin`.

Expand table

Parameter	Description
`/?` or `help`	Displays the syntax and parameters for the `witadmin` command-line tool.
`command /help` or `command /?`	Displays the syntax and parameters for the `witadmin` command that you specify.

XML definition files

You customize all work tracking objects—such as, work item types, process configuration, global lists—by updating their XML definitions. If you're new to work tracking customization, see Customize your work tracking experience.

You run **witadmin** commands against Azure DevOps Services or an on-premises Azure DevOps Server. As shown in the following command lists, only **witadmin** commands that list or export files are supported for the Azure DevOps Services.

Work item types

Work items are used to track anything you need to track. Each work item is based on a work item type. Each work item type defines the fields available in which to store information. The available work item types depend on the process you used when your project was created (Agile, Scrum, Basic, or CMMI). For more information, see Track your work items in Azure Boards user stories, issues, bugs, features, and epics.

For the Inherited process model, you can add and delete work item types from the web portal. For more information, see Add and manage work item types (Inheritance process).

Command

Azure DevOps Services

Azure DevOps Server

`destroywitd`: Delete a work item type

✔

`exportwitd`: Export a work item type file





`importwitd`: Import a work item type file



`listwitd`: List work item types





`renamewitd`: Change the display name for a work item type



Fields

Fields are used to track the status and information associated with work items. Fields are defined for an organization or collection. For more information, see Work item fields and attributes. To look up the description of a specific field, see Field descriptions for default and work item fields used in process templates.

For the Inherited process model, you can view and delete fields from the web portal. For more information, see Add and manage fields (Inheritance process).

Command

Azure DevOps Services

Azure DevOps Server

`deletefield`: Delete a field



`listfields`: List field attributes





`changefield`: Change a field attribute



`renamewitd`: Change a WIT display name



Categories

Categories associate one or more work item types as belonging to the same category. Azure Boards relies on default category definitions, many of which specify a single work item type per category. For more information, see Use categories to group work item types.

Command

Azure DevOps Services

Azure DevOps Server

`exportcategories`: Export the Categories file





`importcategories`: Import the Categories file



Link types

Use different link types to manage the various relationships between work items. For On-premises XML and Hosted XML process models, you can define custom link types. For more information, see Link work items to other objects and Reference guide for link types.

Command

Azure DevOps Services

Azure DevOps Server

`deletelinktype`: Delete a link type

✔

`exportlinktype`: Export a link type file

✔

✔

`listlinktypes`: List the defined link types

✔

✔

`reactivatelinktype`: Reactivate or rename a link type

✔

Global lists

Global lists are pick lists that you can include within one or more fields and work item type definitions. You can share list items among several work item types for a collection of projects by including the list items in one or more **GLOBALLIST** elements. For more information, see GLOBALLIST XML element reference.

Command

Azure DevOps Services

Azure DevOps Server

`destroygloballist`: Delete a global list

✔

`exportgloballist`: Export the global list

✔



`importgloballist`: Import the global list



`listgloballist`: List all global list names





Process configuration

Process configuration defines the default configuration and functional capabilities that your teams can access using Azure Boards. The configuration determines the work item types that appear on the product backlog, sprint backlogs, Kanban board, and task board. For more information about process configuration and what you can customize, see Process Configuration.

For the Inherited process model, you can customize backlogs from the web portal. For more information, see Customize your backlogs or boards (Inheritance process).

Command

Azure DevOps Services

Azure DevOps Server

`exportprocessconfig`: Export the ProcessConfiguration file





`importprocessconfig`: Import the ProcessConfiguration file



Related articles

- Customize your work tracking experience

- On-premises XML process model
- Hosted XML process model

Feedback

Was this page helpful? Yes No

Provide product feedback

Import, export, and manage work item types

Article • 02/21/2023

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

> **ⓘ Important**
>
> For the On-premises XML process model, you can use **witadmin** to list, import, export, and modify a process for a project. For the Inherited and Hosted XML process models, you can only use **witadmin** to list and export process information. For an overview of process models and what is supported, see **Customize your work tracking experience**.

You can manage work item types for a project by using the following `witadmin` commands:

- `destroywitd`: Destroys a work item type, and destroys every work item of that type permanently without recovery.
- `exportwitd`: Exports the definition of a work item type to an XML file, or to the Command Prompt window.
- `importwitd`: Imports work item types from an XML definition file into a project. If a work item type with the same name already exists, the new work item type definition overwrites the existing one. If the work item type doesn't exist, a new work item type is created. To validate the XML that defines a work item type, but not import the file, use the `/v` option.
- `listwitd`: Displays the names of the work item types in the specified project in the Command Prompt window.
- `renamewitd`: Changes the display name of a work item type within a specific project. After you run this command, work items of this type show the new name.

To learn more about how work item types are used to track work, see Track your work items in Azure Boards user stories, issues, bugs, features, and epics.

How to run the `witadmin` command-line tool

To run the `witadmin` command-line tool, open a Command Prompt window where Visual Studio is installed. The `witadmin` command-line tool installs with any version of Visual Studio. You can access this tool by installing the free version of Visual Studio Community or Visual Studio Team Explorer.

> **Note**
>
> If you are connecting to Azure DevOps Services, we recommend that you use the latest version of Visual Studio or Visual Studio Community.

For Visual Studio 2022

`%programfiles(x86)%\Microsoft Visual Studio\2022\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `Professional` or `Enterprise` in place of `Community`, depending on the version you've installed.

For Visual Studio 2019

`%programfiles(x86)%\Microsoft Visual Studio\2019\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `Professional` or `Enterprise` in place of `Community`, depending on the version you've installed.

For Visual Studio 2017

`%programfiles(x86)%\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `TeamExplorer` or `Professional` in place of `Enterprise`, depending on the version you've installed.

On a 32-bit edition of Windows, replace **%programfiles(x86)%** with **%programfiles%**. You can install Visual Studio Community (which provides access to Team Explorer) or Visual Studio Team Explorer 2017 for free.

> **Tip**

With **witadmin**, you can **import and export definition files**. Other tools you can use include the Process Editor (requires that you have installed a version of Visual Studio). **Install the Process Template editor from the Visual Studio Marketplace**.

Prerequisites

For the project where the work item types are defined, you must have the following permissions set:

- To export or list work item types, you must be a member of the **Project Administrators** group or have your **View project-level information** permission set to **Allow**.
- To destroy, import, or rename work item types, you must be a member of the **Team Foundation Administrators** security group or the **Project Administrators** security group.

For more information, see Change project collection-level permissions.

> **Note**
>
> Even if you sign in with administrative permissions, you must open an elevated Command Prompt window to perform this function on a server that is running Windows Server 2008. To open an elevated Command Prompt window, choose **Start**, open the shortcut menu for the **Command Prompt**, and then choose **Run as Administrator**. For more information, see the Microsoft Web site: **User Access Control**.

Syntax

```
witadmin destroywitd /collection:CollectionURL /p:Project /n:TypeName
[/noprompt]
witadmin exportwitd /collection:CollectionURL /p:Project /n:TypeName
[/f:FileName] [/e:Encoding] [/exportgloballists]
witadmin importwitd /collection:CollectionURL [/p:Project] /f:FileName
[/e:Encoding] [/v]
witadmin listwitd /collection:CollectionURL /p:Project
witadmin renamewitd /collection:CollectionURL /p:Project /n:TypeName
/new:NewName [/noprompt]
```

Parameters

Parameter	Description
`/collection`:`CollectionURL`	Specifies the URI of the project collection. For example: **On-premises format:** `http://ServerName:Port/VirtualDirectoryName/CollectionName` If no virtual directory is used, then use the following format: `http://ServerName:Port/CollectionName`.
`/p:Project`	The project for which the types of work items are to be managed. This project must be defined in the project collection specified by the **/collection** parameter. The **/p** parameter is required unless you run the **importwitd** command with the **/v** option.
`/n:TypeName`	The name of the work item type to destroy, export, import, or rename.
`/f:FileName`	The path and file name of the XML definition file that contains the types of work items to be exported or imported. If you omit this parameter when you use the **exportwitd** command, the XML appears in the Command Prompt window. **Note:** If you are using Windows Vista you might not have permissions to certain folders. If you try to export the work item type to a location where you do not have permissions, the registry virtualization technology automatically redirects the exported file and saves it to the virtual store. To avoid this redirection, you can export the file to a location where you have permissions. For more information, see the Registry Virtualization page on the Microsoft web site.
`/e:*Encoding`	The name of a .NET Framework 2.0 encoding format. The command uses the specified encoding to export or import the XML data. For example, `/e:utf-7` specifies Unicode (UTF-7) encoding. If you omit this parameter, **witadmin** tries to detect the encoding, and if detection fails, **witadmin** uses UTF-8.
`/exportgloballists`	Exports the definitions of global lists referenced by the work item type. The definitions for global lists will be embedded into the work item type definition XML. When not specified, the definitions for global lists are omitted.
`/v`	Validates the XML that defines the work item type, but doesn't import the XML definition file. **Note:** You can validate the type definition without specifying a project. References to project-scoped groups is ignored.

Parameter	Description
`/new:NewName`	The new name of the work item type.
`/noprompt`	Disables the prompt for confirmation.
`/?` or `help`	Displays help about the command in the Command Prompt window.

Remarks

When you use the `destroywitd` command, it destroys all the following objects:

- The work item type
- All work items of that type
- Corresponding entries in the work item tables, the long text tables, and the link tables
- Objects in the work item type metadata cache

Examples

Unless otherwise specified, the following values apply in each example:

- URI for the project collection: `http://AdventureWorksServer:8080/tfs/DefaultCollection`
- Project name: `AdventureWorks`
- Input or output file name: `myworkitems.xml`
- Work item type name: `myworkitem`
- Default encoding: `UTF-8`

Export the definition of a WIT

The following command exports the definition for myworkitem to the file, myworkitems.xml.

```
witadmin exportwitd
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/p:AdventureWorks /f:myworkitems.xml /n:myworkitem
```

The following example exports the work item by using Unicode (UTF-7) encoding.

```
witadmin exportwitd
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/p:AdventureWorks /f:myworkitems.xml /n:myworkitem /e:utf-7
```

Export the definition of a WIT and its referenced global lists

The following example exports the work item type and its referenced global lists.

```
witadmin exportwitd
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/p:AdventureWorks /f:myworkitems.xml /n:myworkitem /exportgloballists
```

List the definition of a WIT

The following example displays the definition of the work item type the Command Prompt window.

```
witadmin exportwitd
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/p:AdventureWorks /n:myworkitem
```

Import the definition of WITs

The following example imports the work item definition from the XML file.

```
witadmin importwitd
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/f:myworkitem.xml /p:AdventureWorks
```

Validate the XML definition of a WIT

The following example validates the XML that defines the work item type but doesn't import the definition.

```
witadmin importwitd
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/f:myworkitem.xml /p:AdventureWorks /v
```

Q & A

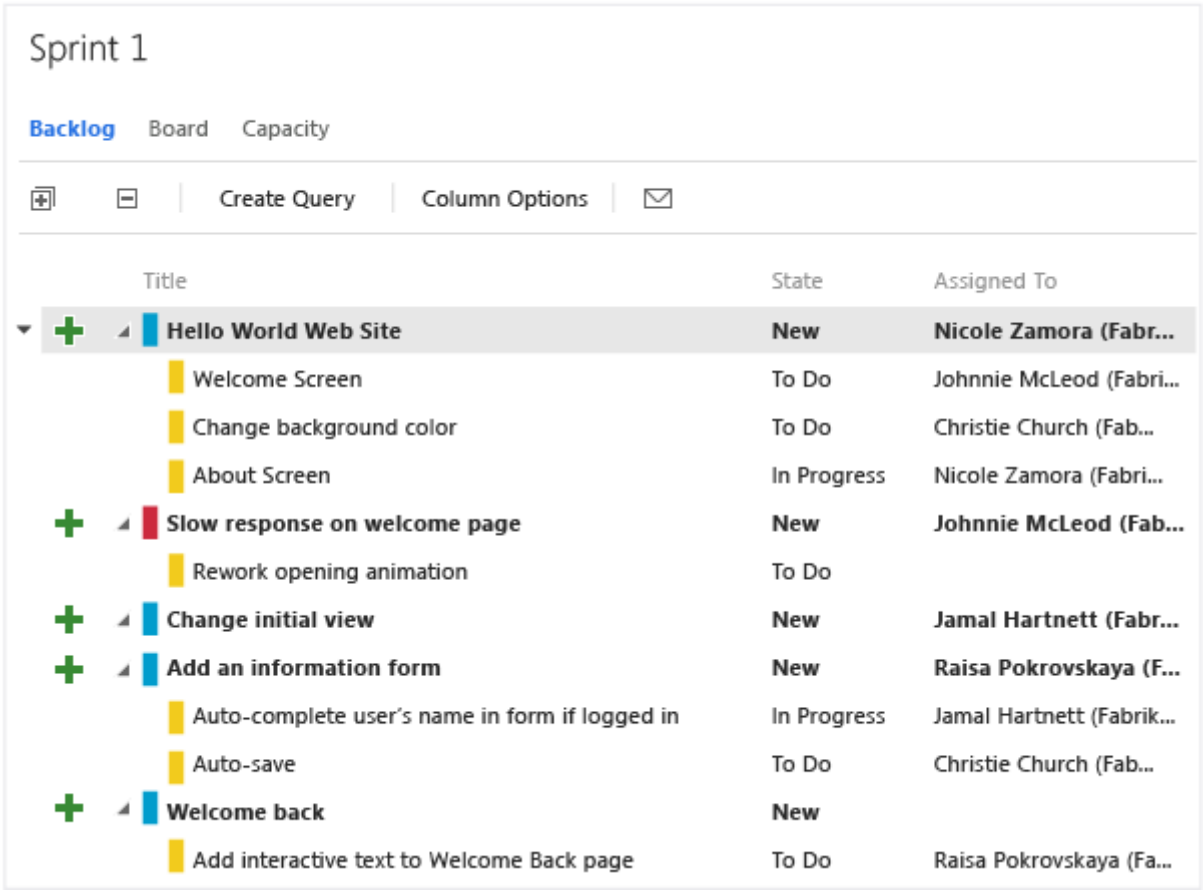
Q: What customizations can I make and still use the Configure Features Wizard to update my project after an upgrade?

A: You can add custom WITs and change the form layout. The Configure Features Wizard will update your projects and you'll get access to the latest features.

Changing the workflow or renaming a WIT might require you to perform some manual operations when updating your project. To learn about which customizations you can safely make and which you should avoid, see Customize the work tracking experience: Before you customize, understand the maintenance and upgrade implications.

Q: How do I change the color associated with a WIT?

A: In the web portal, work items appear in query results and on the backlog and board pages of the Agile planning tools. To change the color associated with an existing WIT or add the color to use for a new WIT, edit the process configuration.



Q: How do I deactivate or disable a WIT? How do I restrict users from creating work items of a certain type?

A: If you have a work item type that you want to retire, but maintain the work items that have been created based on that type, you can add a rule that disables all valid users from saving the work item type.

XML

```
<TRANSITION from=" " to="New">
   <FIELDS>
     <FIELD refname="System.CreatedBy">
          <VALIDUSER not="[Team Project Name]Project Valid Users" />
     </FIELD>
   </FIELDS>
</TRANSITION>
```

If you want to restrict creation of a specific WIT to a group of users, there are two ways to restrict access:

- Add the WIT to the Hidden Categories group to prevent the majority of contributors from creating them. If you want to allow a group of users access, you can create a hyperlink to a template that opens the work item form and share that link with those team members who you do want to create them.

- Add a field rule to the workflow for the System.CreatedBy field to effectively restrict a group of users from creating a work item of a specific type. As the following example shows, the user who creates the work item must belong to the `Allowed Group` in order to save the work item.

XML

```
<TRANSITION from=" " to="New">
   <FIELDS>
     <FIELD refname="System.CreatedBy">
         <VALIDUSER for="Allowed Group" not="Disallowed Group" />
     </FIELD>
   </FIELDS>
</TRANSITION>
```

Q: How do I delete a WIT?

A: To prevent team members from using a specific WIT to create a work item, you can remove it from the project. When you use `witadmin destroywitd`, you permanently remove all work items that were created using that WIT as well as the WIT itself. For example, if your team doesn't use Impediment you can delete the WIT labeled Impediment from the Fabrikam Web Site project.

```
witadmin destroywitd
/collection:"http://FabrikamPrime:8080/tfs/DefaultCollection" /p:"Fabrikam
Web Site" /n:"Impediment"
```

When you delete a WIT that belongs to a category, you must update the categories definition for the project to reflect the new name. In particular, the Agile planning tools will not work until you update the categories definition.

For more information, see Import and export categories.

Related articles

- Customize your work tracking experience
- Work item field index
- witAdmin: Customize and manage objects for tracking work

Manage work item fields

Article • 03/25/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

> **ⓘ Important**
>
> For the On-premises XML process model, you can use **witadmin** to list, import, export, and modify a process for a project. For the Inherited and Hosted XML process models, you can only use **witadmin** to list and export process information. For an overview of process models and what is supported, see Customize your work tracking experience.

You can manage the fields defined for work item types that are defined for a project collection (On-premises XML) by using the following **witadmin** commands. If you want to add a global field (valid for On-premises XML) you can do so by modifying the global workflow file and importing it to the collection.

- `changefield`: Changes one or more attributes of a field. When you change one of the following attributes, you change it for all work item types and projects within the project collection:
  - **Data type** for `PlainText` or `HTML` fields.

    > **ⓘ Important**
    >
    > When you upgrade Team Foundation Server from an earlier version to the current version, the type assignment for the **Description** (System.Description) field is automatically converted from `PlainText` to `HTML`. With the `changefield` command, you can restore the content of this field to display plain text.

  - **Friendly name** that displays in the work item query. This name may differ from that displayed on the work item form.
  - **Reporting attributes** which includes the name of the field as it appears in a report, the reference report name, and the reporting type.
  - **Synchronization** with Active Directory - you can enable/disable synchronization of person name fields.

- `deletefield`: Deletes the specified field.
- `listfields`: Lists the attributes for all fields or a specified field.

> ⓘ Note
>
> The `witadmin indexfield` command has been deprecated with Azure DevOps Server 2019 and later versions. Indexing fields is no longer required.

Run the `witadmin` command-line tool

To run the `witadmin` command-line tool, open a Command Prompt window where Visual Studio is installed. The `witadmin` command-line tool installs with any version of Visual Studio. You can access this tool by installing the free version of Visual Studio Community or Visual Studio Team Explorer.

> ⓘ Note
>
> To connect to Azure DevOps Services, we recommend that you use the latest version of Visual Studio or Visual Studio Community.

For Visual Studio 2022

`%programfiles(x86)%\Microsoft Visual Studio\2022\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `Professional` or `Enterprise` in place of `Community`, depending on the version you installed.

For Visual Studio 2019

`%programfiles(x86)%\Microsoft Visual Studio\2019\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `Professional` or `Enterprise` in place of `Community`, depending on the version you installed.

For Visual Studio 2017

`%programfiles(x86)%\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `TeamExplorer` or `Professional` in place of `Enterprise`, depending on the version you installed.

On a 32-bit edition of Windows, replace **%programfiles(x86)%** with **%programfiles%**. You can install Visual Studio Community (which provides access to Team Explorer) or Visual Studio Team Explorer 2017 for free.

For an overview of the fields defined within a default process template, see Work item field index.

> **Tip**
>
> With **witadmin**, you can **import and export definition files**. Other tools you can use include the Process Editor (requires that you have installed a version of Visual Studio). **Install the Process Template editor from the Visual Studio Marketplace**.

Prerequisites

- To list fields, you must have your **View project-level information** permission for the project in the collection set to **Allow**.
- To delete or rename fields or change an attribute of a field, you must be a member of the **Team Foundation Administrators** security group or the **Project Collection Administrators** security group.

For more information, see Change project collection-level permissions.

> **Note**
>
> Even if you sign in with administrative permissions, you must open an elevated Command Prompt window to perform this function on a server that is running Windows Server 2008. To open an elevated Command Prompt window, choose **Start**, open the **Command Prompt** shortcut menu, and then choose **Run as Administrator**. For more information, see the Microsoft Web site: **User Access Control**.

Syntax

```
witadmin changefield /collection:CollectionURL /n:RefName   [/name:NewName]
[/syncnamechanges:true | false]   [/reportingname:ReportingName]
[/reportingrefname:ReportingRefName]   [/reportingtype:Type]
[/reportingformula:Formula]   [/type:PlainText | HTML]   [/noprompt]

witadmin deletefield /collection:CollectionURL /n:RefName [/noprompt]

witadmin listfields /collection:CollectionURL /n:RefName [/unused]
```

Parameters

Expand table

Parameter	Description
`/collection`:`CollectionURL`	Specifies the URI of the project collection. For example: **On-premises format:** `http://ServerName:Port/VirtualDirectoryName/CollectionName` If no virtual directory is used, then use the following format: `http://ServerName:Port/CollectionName`.
`/n:RefName` `/n:Name`	The reference name of a work item type field.
`/index`	Specifies to enable or disable indexing for the specified field. Specify **on** to enable indexing and **off** to disable indexing.
`/name:NewName`	Specifies the new name for the field.
`/syncnamechanges`	Specifies to use the work item field to store names and to update as changes are made in Active Directory or a workgroup. This option is valid only when a field with the data type of String is specified for the `typename`. Specify `true` to enable synchronization for the data field, specify `false` to disable synchronization for the data field.
`/reportingname:ReportingName`	Specifies the name of the field in the data warehouse to be used for reporting.
`/reportingrefname:ReportingRefName`	Specifies the reference name of the field in the data warehouse to be used for reporting.

Parameter	Description
`/reportingtype:Type`	Specifies how the field is used in the warehouse for reporting. The following values are valid: - `dimension:` Used for the Integer, String, or DateTime fields. - `detail:` Used for the Integer, Double, String, or DateTime fields. - `measure:` Used for the Integer and Double fields. The default aggregation type is sum. You can specify another aggregation type by using the **formula** parameter. - `none:` Used to disable reportability on the field. For more information, see About work item fields and attributes.
`/reportingformula:Formula`	Specifies the aggregation formula to be used when the field is reported as a `measure`. The only supported formula is `sum`.
`/type:HTML` | `PlainText`	Specifies to convert the contents of the field from `PlainText` to `HTML` or from `HTML` to `PlainText`. You can specify this option only for fields whose type assignment is `PlainText` or `HTML`. See FIELD (Definition) element reference.
`/unused`	Lists all fields that are not used by any project defined in the project collection.
`/noprompt`	Disables prompt for confirmation.
`/?` or `help`	Displays help about the command in the Command Prompt window.

Synchronizing person names with Active Directory

You must manually enable synchronization of any custom work item fields that are used to assign person names that reference Active Directory. You must enable synchronization for each field for each project collection that contains the custom fields.

All system reference fields that show person-names have the attribute `syncnamechanges` set to `true`. Such fields include System.AuthorizedAs, System.AssignedTo, System.ChangedBy and System.CreatedBy. Synchronization in enabled for each person name field that is defined in one of the default process templates. For more information, see Assignments and workflow fields.

After synchronization is enabled, the field no longer shows a static string. Instead, the field shows the name associated with a user account. When you change the user name

in Active Directory or in Workgroup, a field with `syncnamechanges` set to `true` automatically shows the new name.

When you assign the `syncnamechanges` attribute to a String field, the field always accepts valid user names. However, the field doesn't allow group names that are stored in Team Foundation Server or in Active Directory if any one of the following conditions is `true`:

- The `VALIDUSER` rule is specified across all work item types
- The `VALIDUSER` rule is specified for a work item type
- The `ALLOWEDVALUES` rule is specified for a work item type, and that rule has a filter criteria that excludes groups

  For more information, see All FIELD elements.

Attributes that you can change for each work item type

You change the following attributes or values defined for a field by changing the work item type definition in which the field appears:

- **Name** that displays on the work item form. See WebLayout and Control elements or Control XML element reference.
- **Help text**. See Rules and rule evaluation.
- **Allowed values** or items within a pick list or drop-down menu. See ALLOWEDVALUES, SUGGESTEDVALUES, and PROHIBITEDVALUES XML elements.

Examples

Unless otherwise specified, the following values apply in each example:

- URI for the project collection: http://AdventureWorksServer:8080/tfs/DefaultCollection
- Work item field name: AdventureWorks.Field
- Default encoding: UTF-8

List fields

Use `witadmin listfields` to see the set of fields in use, to select one to add to a work item type. Also, you can list the attribute assignments defined for a specific field and determine which fields are used by which projects.

View the attributes of a work item field

- Enter the following command to list the attributes defined for a specified field, such as `Microsoft.VSTS.Common.Issue`.

```
witadmin listfields
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/n:Microsoft.VSTS.Common.Issue
```

Field and attribute information appears for the named field, as shown in this example.

```
Field: Microsoft.VSTS.Common.Issue
Name: Issue
Type: String
Reportable As: dimension
Use: Adventure Works (Shared Steps), AW Future (Shared Steps), AW
Current (Shared Steps)
Indexed: False
```

The **Use** parameter indicates the name of each project and the work item type where the field is used. For more information about field attributes, see Index of work item fields.

List all fields in a project collection

- Enter the following command to list all fields defined for a project collection.

```
witadmin listfields
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
```

Field information for all the fields for the named project collection appears. See Index of work item fields.

List fields that are not being used

- Enter the following command to list the fields that are no longer being used in the project collection by any work item type.

```
witadmin listfields
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/unused
```

Field and attribute information appears for each field that is not being used, as shown in this example.

```
Field: Microsoft.VSTS.CMMI.TaskType
Name: Task Type
Type: String
Reportable As: dimension
Use: Not In Use
Indexed: False

Field: Microsoft.VSTSUE.Common.Flag
Name: Flag
Type: String
Reportable As: dimension
Use: Not In Use
Indexed: False

Field: Microsoft.VSTSUE.Common.Progress
Name: Progress
Type: String
Reportable As: dimension
Use: Not In Use
Indexed: False
```

Rename a field

You can change the friendly name of a work item field to meet the naming conventions that your team uses. Note that the new name is applied to all work item types that reference the changed field in all projects in the project collection. The friendly name displays when you define filter criteria in a work item query. The name that appears on a work item form may be different than the friendly name defined for the field.

To rename a work item field

1. Enter the following command to rename the friendly name assigned to `Microsoft.VSTS.Common.Rank` to **Important Rank**.

```
```

```
witadmin changefield
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/n:Microsoft.VSTS.Common.Rank /name:"Important Rank"
```

2. At the confirmation prompt type **y** and then press ENTER. Wait until you receive the following message:

   Field renamed.

To verify changes imported to a single project

1. In Team Explorer, choose **Refresh**.

   The latest updates are downloaded from the server, which include the changes that you just made to the field name. Wait several seconds for the refresh to finish.

2. Choose **New Query** to create a query.

3. In the Query Editor, choose the **Click here to add a clause** link to add a row, select the blank **Field** cell, and in the cell, type **Rank**. The following message that appears above the results list. This message indicates that the Rank cannot be found.

   Run the query to see the query results. TF51005: The query references a field that doesn't exist. The error is caused by <<Rank>>.

4. Delete the value **Rank** from the **Field** cell, and type **Important Rank** into the cell.

5. Select <> in the **Operator** cell, and type **1** into the **Value** cell.

6. On the **Query** toolbar, choose **Run**.

7. Open the shortcut menu for any row in the results and select **Column Options**. Scroll down in the **Available columns** list. Notice that the **Rank** field is no longer present but the **Important Rank** field is present.

8. Select **Important Rank** in the **Available columns** box, and then choose the > button (add selected columns). Choose **OK**.

   Notice that the friendly name for Microsoft.VSTS.Common.Rank has been renamed from **Rank** to **Important Rank** throughout the query builder and results list.

9. Close the query. Choose **No** when you are prompted to save the query.

10. Create a new Task work item. Choose the **New Work Items** link, and then choose **Task**.

This work item is created from the work item type that you changed and imported.

11. Notice, in the **Status** box, that the label for the renamed field, **Rank**, has not changed. This is because the field labels on the work item forms are scoped to the parent project and are independent of the server-wide field name just specified.

    > ⓘ **Note**
    >
    > For more information about how to change field labels on work item forms, see Control XML element reference.

12. Close the new Task and choose **No** when you are prompted to save the work item.

Change the report as value for a field

The following command specifies the ability to report the type of the DateTime field AdventureWorks.CreatedOn to dimension. This field's data enters the warehouse and Analysis Services databases so that it can be used to filter reports.

```
witadmin changefield
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/n:AdventureWorks.CreatedOn /reportingtype:dimension
```

The following command specifies the ability to report the type of the Double field AdventureWorks.Field to measure. All measures are aggregated by sum.

```
witadmin reportfield
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/n:AdventureWorks.Field /reportingtype:measure
```

Enable synchronization of a custom person-name field

The following command enables synchronization for the work item field AW.CustomerName defined for Collection1 on the AdventureWorksServer.

Verify the data type of the field that you want to convert

1. Verify the data type assigned to the field, such as MyCompany.CustomerName, that you want to synchronize by entering the following command:

```
witadmin listfields
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/n:MyCompany.CustomerName
```

Enable synchronization

1. To enable synchronization for a person-named field, type the following command, substituting your data for the arguments that are shown here:

```
witadmin changefield
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/n:MyCompany.CustomerName /syncnamechanges:true
```

2. The following confirmation prompt appears:

   This will change properties for field {0} on the Team Foundation Server. Do you want to continue?

3. Type **0** to confirm that you want to change the field, or **1** to cancel the request.

   If the change request succeeds, the following confirmation message appears:

   The field was updated.

   If the change request fails, an error message appears. The most common mistakes that can be made are trying to change a system reference field, or trying to change a field of a data type other than String. These operations are not supported.

Delete a field

Before you delete a field, verify that the field is not in use. If the field is in use, you must first remove the field from the work item types that use it prior to deleting it from the project collection. The following command deletes the field `AdventureWorks.Field` from *Collection1*:

```
```

```
witadmin deletefield
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/n:AdventureWorks.Field
```

Verify a field is not in use

1. Enter the following command, specifying the reference name for the work item field, such as `MyCompany.MyProcess.MyField`.

   ```
   witadmin listfields
   /collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
   /n:MyCompany.MyProcess.MyField
   ```

   In the information displayed for the field, verify that the value for **Use** is "**Not In Use**" as follows in this example.

   ```
   Field: MyCompany.MyProcess.MyField
   Name: MyField
   Type: String
   Reportable As: dimension
   Use: Not In Use
   Indexed: False
   ```

2. If the **Use** field indicates that the field is in use, then you must delete it from each work item type for each project that is listed. For example, the `Microsoft.VSTS.TCM.SystemInfo` field indicates that it is being used by the Bug and Code Defect work item types for four projects: Arroyo, Desert, Palm, and Springs.

   ```
   Field: Microsoft.VSTS.TCM.SystemInfo
   Name: System Info
   Type: Html
   Reportable As: None
   Use: Arroyo (Bug), Desert (Bug), Palm (Bug), Springs (Bug, Code Defect)
   Indexed: False
   ```

   Before you can delete this field, you must remove it from each of the work item types listed for each project for which it is defined. To remove the field, you modify the definition for the work item type by deleting the `FIELD` and `Control` elements

that contain the field reference name. See Import, export, and manage work item types, FIELD (Definition) element reference, and Control.

Delete a field from a project collection

Enter the following command to delete the `MyCompany.MyProcess.MyField` field, and then choose Enter.

```
witadmin deletefield
/collection:http://AdventureWorksServer:8080/tfs/DefaultCollection
/n:RefName
```

Enter **y** at the confirmation prompt to complete this step.

Related articles

- Customizing your work tracking experience
- Work item field index
- witAdmin: Customize and manage objects for tracking work

Feedback

Was this page helpful? Yes No

Provide product feedback

Import and export process configuration

Article • 03/25/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

> **Important**
>
> For the On-premises XML process model, you can use **witadmin** to list, import, export, and modify a process for a project. For the Inherited and Hosted XML process models, you can only use **witadmin** to list and export process information. For an overview of process models and what is supported, see Customize your work tracking experience.

You customize the process configuration to modify the display of the web portal Agile tools. A few additional tools require that you map workflow states to metastate mappings.

> **Note**
>
> If you receive error message TF400917, an invalid configuration has occurred. Re-import the process configuration file to your project using `witadmin importprocessconfig`. You'll get the messages you need to resolve the error.

To manage the process configuration for a project, use the `witadmin` command line tool to export and import the process configuration XML definition file. To learn about process configuration, see ProcessConfiguration.

- `exportprocessconfig`: Exports the process configuration definition to an XML file or the Command Prompt window.
- `importprocessconfig`: Imports the process configuration definition XML file.

Run the `witadmin` command-line tool

To run the `witadmin` command-line tool, open a Command Prompt window where Visual Studio is installed. The `witadmin` command-line tool installs with any version of Visual Studio. You can access this tool by installing the free version of Visual Studio Community or Visual Studio Team Explorer.

> **Note**
>
> To connect to Azure DevOps Services, we recommend that you use the latest version of Visual Studio or Visual Studio Community.

For Visual Studio 2022

`%programfiles(x86)%\Microsoft Visual Studio\2022\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `Professional` or `Enterprise` in place of `Community`, depending on the version you installed.

For Visual Studio 2019

`%programfiles(x86)%\Microsoft Visual Studio\2019\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `Professional` or `Enterprise` in place of `Community`, depending on the version you installed.

For Visual Studio 2017

`%programfiles(x86)%\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer`

or `TeamExplorer` or `Professional` in place of `Enterprise`, depending on the version you installed.

On a 32-bit edition of Windows, replace **%programfiles(x86)%** with **%programfiles%**. You can install Visual Studio Community (which provides access to Team Explorer) or Visual Studio Team Explorer 2017 for free.

> **Tip**
>
> With **witadmin**, you can **import and export definition files**. Other tools you can use include the Process Editor (requires that you have installed a version of Visual Studio). **Install the Process Template editor from the Visual Studio Marketplace**.

Prerequisites

- To export process configuration definitions, you must be a valid user of the project or collection.
- To import process configuration definitions, you must be a member of the following security groups: **Team Foundation Administrators** or **Project Administrators**.

For more information, see Change project collection-level permissions.

Syntax

```
witadmin exportprocessconfig /collection:CollectionURL /p:ProjectName
[/f:FileName] [/e:encoding]
witadmin importprocessconfig /collection:CollectionURL [/p:ProjectName]
/f:FileName [/e:encoding] [/v]
```

Parameters

Parameter

Description

`/collection:CollectionURLx`

Specifies the URI of the project collection. For example:

On-premises Azure DevOps format:
`http://ServerName:Port/VirtualDirectoryName/CollectionName`

If no virtual directory is used, then the format for the URI is the following: `http://ServerName:Port/CollectionName`.

`/p:ProjectName`

Required. Specifies the project for which you want to export or import the process configuration. This project must be defined in the collection that you specified by using the /collection parameter. You do not need to specify a project when you specify the `/v` switch.

`/f**:FileName`

The path and the name of the XML definition file for the process configuration to export or import.

> **Note**
>
> If the client computer is running Windows Vista, you might not have permissions to certain folders. If you try to export the global list to a location where you do not have permissions, the registry virtualization technology automatically redirects the exported file and saves it to the virtual store. For more information, see the Registry Virtualization page on the Microsoft website. To avoid this redirection, you can export the file to a location where you have permissions.

`/e:Encoding`

Optional. The name of a .NET Framework 2.0 encoding format. The specified encoding will be used to export or import the XML data. For example, `/e utf-7` specifies Unicode (UTF-7) encoding. If you omit this parameter, **witadmin** attempts to detect the encoding and uses UTF-8 if detection fails.

`/v`

Optional. Validates the XML that defines the process configuration but doesn't import the definition file.

`/?` or `help`

Displays help about the command in the Command Prompt window.

Remarks

Installing the latest version of Azure DevOps upgrades existing projects. If you are updating an upgraded project, see Configure features after an upgrade.

If you encounter problems accessing existing test plans or test suites after an upgrade, see Manual updates to support test management.

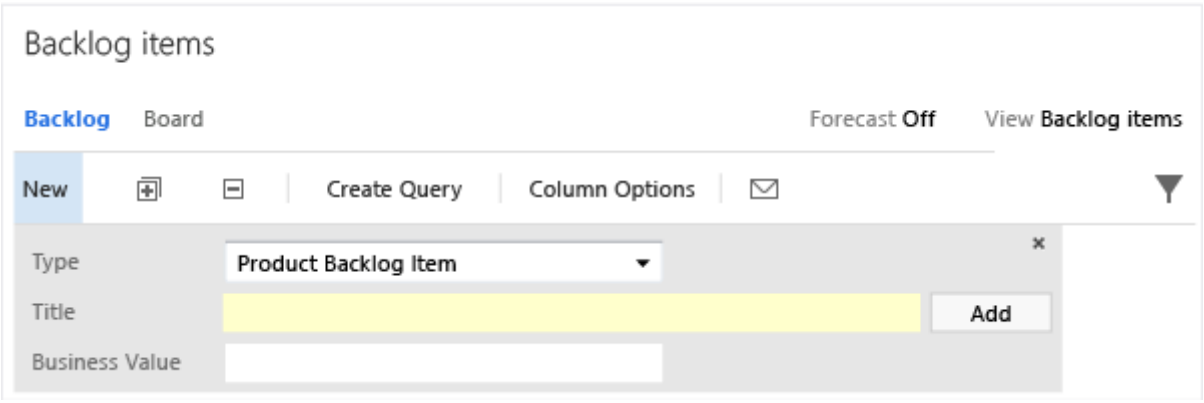
Examples

The following values apply in each example:

- URL for the collection: `http://AdventureWorksServer:8080/tfs/DefaultCollection`
- Project: `Contoso`
- Port number for the server website: `8080`

To add a field to the quick add panel

You can add fields for any quick add panel. For example, the following example adds **Business Value** to the product backlog panel.



The panel only displays fields that are included in the `FIELDS` section of the WIT definition for the WIT selected. For example, if you select the bug WIT, then only Title displays, as Business Value isn't defined for bugs. To add another WIT to the panel, you add it to the Requirements Category. See Add a work item type to a backlog and board.

1. If you don't have administrative permissions for your project, get them.
2. Open a Command Prompt window according to the instructions provided in How to run the witadmin command-line tool. For example:

```
cd %programfiles(x86)%\Microsoft Visual Studio\2017\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\Team Explorer
```

3. Export the process configuration file.

```
witadmin exportprocessconfig /collection:CollectionURL /p:ProjectName
```

```
/f:"DirectoryPath\ProcessConfiguration.xml"
```

An example of a *CollectionURL* is http://MyServer:8080/tfs/TeamProjectCollectionName.

4. Add the field reference name to the `AddPanel` section.

   For example, the following syntax adds the **Priority** element to the panel.

   XML

   ```
   <AddPanel>
      <Fields>
         <Field refname="System.Title" />
         <Field refname="Microsoft.VSTS.Common.BusinessValue" />
      </Fields>
   </AddPanel>
   ```

   Tip

   - You can look up the reference name for a field using this **index**.
   - Add all required fields for work item types defined for the Requirements Category. That way, you avoid having to open the work item form to fill them in when you add backlog items through the panel.

5. Import the process configuration file.

   ```
   witadmin importprocessconfig /collection:CollectionURL /p:ProjectName
   /f:"DirectoryPath\ProcessConfiguration.xml"
   ```

6. Refresh your backlog page to view the updated panel.

Update metastate mappings to support Test Manager

If you customize the `WORKFLOW` section of the test plan or test suite, you must map the states to metastates.

In the following example, the test plan workflow has been updated to use the Design, Testing, and Signed Off states. To support backward compatibility, the

`TestPlanWorkItems` is added to the `ProjectProcessConfiguration` section of the process configuration definition.

Workflow

XML

```
<WORKFLOW>
      <STATES>
        <STATE value="Design" />
        <STATE value="Testing" />
        <STATE value="Signed Off" />
      </STATES>
      <TRANSITIONS>
        <TRANSITION from="" to="Design">
          <REASONS>
            <DEFAULTREASON value="New test plan" />
          </REASONS>
        </TRANSITION>
     <TRANSITION from="Design" to="Testing">
          <REASONS>
            <DEFAULTREASON value="Authoring complete" />
          </REASONS>
        </TRANSITION>
     <TRANSITION from="Testing" to="Signed Off">
          <REASONS>
            <DEFAULTREASON value="Signed Off testing" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Signed Off" to="Design">
          <REASONS>
            <DEFAULTREASON value="Reactivating to authoring phase" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Signed Off" to="Testing">
          <REASONS>
            <DEFAULTREASON value="Reactivating to run tests" />
          </REASONS>
        </TRANSITION>
        <TRANSITION from="Testing" to="Design">
          <REASONS>
            <DEFAULTREASON value="Back to authoring" />
          </REASONS>
        </TRANSITION>
      </TRANSITIONS>
    </WORKFLOW>
```

Metastate mappings

XML

```
<TestPlanWorkItems category="Microsoft.TestPlanCategory" pluralName="Test
Plans" singularName="Test Plan">
    <States>
      <State type="InProgress" value="Design" />
      <State type="InProgress" value="Testing" />
      <State type="Complete" value="Signed Off" />
    </States>
  </TestPlanWorkItems>
<TestSuiteWorkItems category="Microsoft.TestSuiteCategory" pluralName="Test
Suites" singularName="Test Suite">
    <States>
      <State type="Proposed" value="In Planning" />
      <State type="InProgress" value="In Progress" />
      <State type="Complete" value="Completed" />
    </States>
  </TestSuiteWorkItems>
```

If you modify the test suite workflow, then you have to make a similar update if you want to map new states. You would add it within a `TestSuiteWorkItems` section. See ProcessConfiguration.

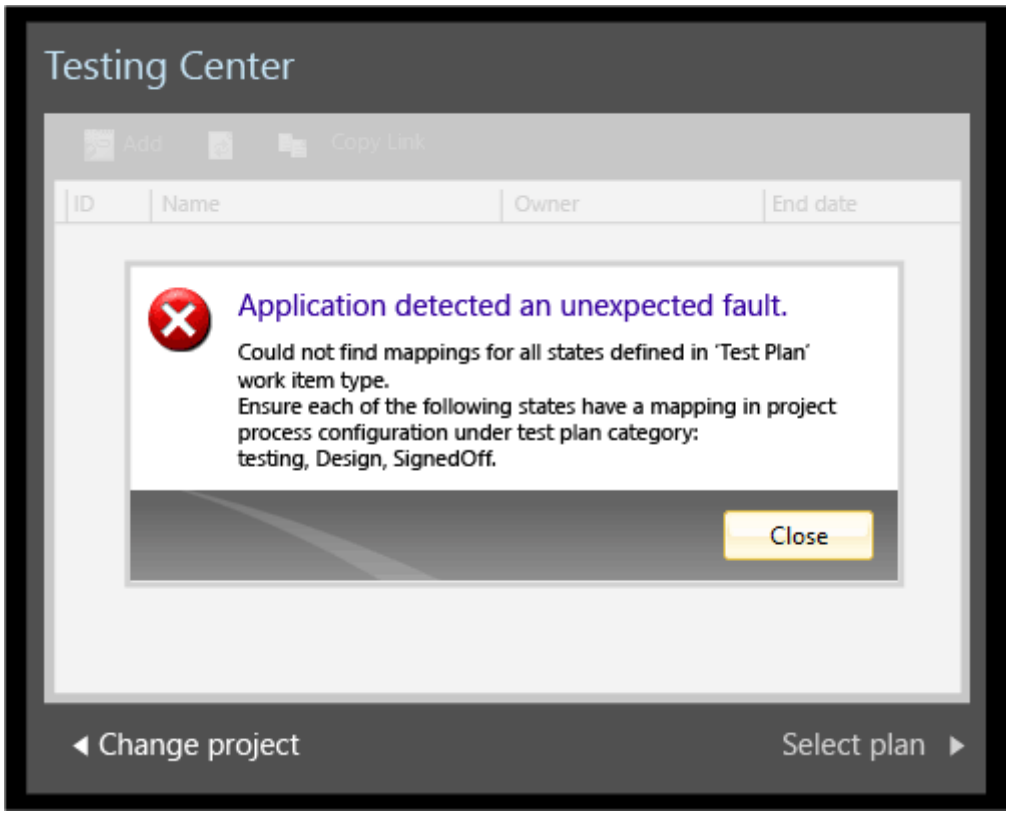
Q & A

Q: When do I need to map workflow states to metastates?

A: When you add or remove workflow states to the following work item types, you should consider updating the process configuration to add or remove corresponding metastate mappings.

- **Work item types that belong to the Requirement Category or Task Category**: Metastate mappings support the display of the Agile planning tools.
- **Work item types that belong to the Bug Category**: Metastate mappings to support **My Work** tool (Agile and CMMI-based projects).
- **Test Plan and Test Suite**: Updates to the workflow of these work item types must be mapped only when you support team members connecting to Azure DevOps from a version of Test Manager that is based on Visual Studio 2013.2 or earlier version.

  Update the metastate mappings if you receive an **Application detected an unexpected fault** error when you connect to your project.



How do I resolve process configuration errors?

A: When you make one of the following changes to a project, you'll need to update the definitions for the work item type, categories, or process configuration. To avoid errors, always make your changes in this order: (1) work item types, (2) categories, and (3) process configuration.

Customization

Update or verify the work item type definition

Update or verify the process configuration definition

Add a work item type to the Requirements Category

(A work item type can belong to the Requirements Category or the Task Category, but not both.)

To include the following fields:

- Backlog Priority (Scrum) or Stack Rank (Agile or CMMI)

  (must match the field value assigned to type=Order in the process configuration file)
- Effort (Scrum), Story Points (Agile), or Size (CMMI)

(must match the field value assigned to `type=Effort` in the process configuration file)
- Area path or the field value assigned to `type=Team` in the process configuration file

- All fields that are included in the `AddPanel` section of the process configuration file (fields must be defined within the `FIELDS` section but don't have to be included within the `FORM` section.

To contain the necessary metastate mappings:
- Map the start of each workflow state to `type="Proposed"`
- Map each intermediate workflow state you want to have show up on the Kanban board to `type="InProgress"`
- Map the end of each workflow state to `type="Complete"`
  You can have only one State mapped to `type="Complete"`

  To contain an entry to define the color codes associated with the WIT. For example: `<WorkItemColor primary="FF009CCC" secondary="FFD6ECF2" name="Product Backlog Item" />`

Change the workflow of a work item type in the Requirements Category

N/A

To contain the necessary metastate mappings as described above for adding a work item type to the Requirements Category.

Add a work item type to the Task Category

To include the following fields:
- Backlog Priority (Scrum) or Stack Rank (Agile or CMMI)
- Activity (Scrum or Agile) or Discipline (CMMI)
  (must match the field value assigned to `type=Activity` in the process configuration file)
- Remaining Work
  (must match the field value assigned to `type=RemainingWork` in the process configuration file)
- Area path or the field value assigned to `type=Team` in the process configuration file
- (Optional) Original Work and Completed Work (Agile and CMMI only)

To contain the necessary metastate mappings:
- Map the start of each workflow state to type="Proposed"

- Map each intermediate workflow state that you want to have show up on the task board to `type="InProgress"`
- Map the end of each workflow state to `type="Complete`"

  You can have only one State mapped to `type="Complete"`

To contain an entry to define the color codes associated with the WIT. For example:

```
<WorkItemColor primary="FFF2CB1D" secondary="FFF6F5D2" name="Task" />
```

Change the workflow of a work item type in the Task Category

N/A

To contain the necessary metastate mappings as described above for adding a work item type to the Task Category.

Add a work item type to the Bug Category (Agile and CMMI only)
Change the workflow of a work item type in the Bug Category (Agile and CMMI only)

N/A

To contain the necessary metastate mappings:

- Map the start of each workflow state to `type="Proposed"`
- Map each intermediate workflow state you want to have show up for My Work to `type="InProgress"`
- Map the end of each workflow state type="Complete"

You can have only one State mapped to `type="Complete"`

To learn more, see Support bug update status using My Work.

Remove a work item type from the Requirements Category or Task Category

N/A

To remove any metastate mappings that are only associated with that work item type

Remove a work item type from a project

To remove the work item type from the categories file.

To remove any metastate mappings that are only associated with the work item type that you removed and the `WorkItemColor` element that defines the color codes for the

work item type you removed.

Q: Do you want to work with two or more portfolio backlogs?

A: The default experience supports one level of portfolio backlog. You can add up to five levels as described in Add portfolio backlogs to Agile tools.

Q: Do you want to add or change the work item types that appear on your task board or product backlog?

A: If you've added a custom work item type and want to add that to either the backlog or task board, you can. You just can't have them appear in both places. Learn how by reading Add work item types to backlogs and boards.

Related articles

- Configure features after an upgrade
- Control XML element reference
- Change the work item form layout
- Edit a work item type definition to add web content to a work item form
- Customizing your work tracking experience
- Work item field index
- witAdmin: Customize and manage objects for tracking work

Feedback

Was this page helpful? Yes No

Provide product feedback

Azure Boards FAQs

FAQ

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Find answers to frequently asked questions about using Azure Boards. For FAQs specific to queries or Microsoft Excel integration to add or modify work items defined in Azure DevOps, see Query FAQs and FAQs: Work in Excel connected to Azure Boards.

You can view a list of features that are on our roadmap for Azure Boards from the Features Timeline. To request a feature or up-vote a feature, go to our Developer Community page.

Access and permissions

Who can contribute to Azure Boards?

As a member of an Azure Boards project, you can use most features to track work. Limitations to select features are based on the access level and security group to which a user is assigned. Other limitations can be imposed based on work tracking permissions or custom rules. For more information, see Default permissions and access for Azure Boards.

How can noncontributors view or modify work items?

For private projects, you can grant access to an unlimited number of users by assigning them Stakeholder access. For public projects, anonymous users—ones who don't sign into the project—can view all work items. For more information, see Stakeholder access quick reference and Make a private project public.
Also, if your organization uses Microsoft Entra ID to manage user access, you can add external users to your organization.

How can I restrict viewing or modifying select work items?

You can restrict access by setting permissions for an area path. For more information, see the following articles:

- Set work tracking and plan permissions
- Grant or restrict permissions to select tasks

Can caching affect rules applied to work items?

Yes. Conditional rules based on user or group membership get cached for your web browser. If you find yourself restricted to update a work item, you probably encountered one of these rules. For more information about conditional rules, see Rules and rule evaluation. If your cache stores outdated rules, you can wait for the client cache to expire within three days, or you can clear the cache.
To clear the cache, run the following command in the browser command window and then refresh the browser: window.indexedDB.deleteDatabase("wit")

What client tools support work tracking with Azure Boards?

See Tools and clients that connect to Azure DevOps.

Work items

Where should I start to learn about work items and work item types?

For an introduction to work items, see About work items.

How do I view all work items?

To view all work items defined in a project, open a query and add two clauses:

- `Work Item Type = [Any]`
- `State = [Any]`

From there, you can add filters to refine the query. For more information, see Create and save managed queries with the query editor.

Can a work item be assigned to several users or a user group?

No. Work items can only be assigned to a single user. Also, you can only add a user that is available from the people picker.

What's the recommended method to group work?

The main method to group work is to assign it to the same area path. Area paths are used to group work items by product, feature, or business areas and to support teams responsible for work assigned to those areas. In addition, you can group work under a parent work item using parent-child links, referred to as a hierarchical grouping.

For a discussion of these two usages and the tools they support, see Configuration and customization of Azure Boards, Area paths, product teams, and portfolio management.

How do I mark a task or work item as a milestone task?

Milestone markers aren't used in Azure Boards work tracking, except for Delivery Plans. Delivery Plans provide a calendar view and allow you to define a milestone marker. For more information, see Review team Delivery Plans.

However, you can use one or more of the following options to mark a work item as a milestone:

- Prepend or append the word **Milestone** in the title of your work item
- Add a work item tag labeled **Milestone**
- Add a custom field labeled **Milestone** and populate it with a pick list of milestones
- Link work items using the Predecessor/Successor or Related link type to a milestone work item
- Assign the milestone work item to the sprint in which it's targeted for completion.

How can I best track dependencies?

You can track dependencies between work items by linking them using a related or other link type. See Link work items to other objects.

You can view dependencies using Delivery Plans.

If you're tracking dependencies across one or more organizations, you might want to use the Dependency Tracker.

What determines which work item type should be a parent for another work item type?

Each process defines four default backlog levels: the iteration backlog, requirement backlog, feature and epic portfolio backlogs. The work item types added to each backlog level are the natural work item types to form parent-child relationship and group work items into a hierarchy. For more information, see Organize your backlog, map child work items to parents. To customize the process, see Customize your backlogs or boards, Edit or rename the requirement backlog.

How do I copy or clone a work item with all linked items?

With Azure Boards (cloud service), you can choose to copy child work items when copying a work item. For more information, see Copy or clone work items.

How do I bulk modify a rich-text field?

See Bulk modify work items, Bulk modify rich-text fields.

Automation rules

Is there way to make the parent active when a child is active?

Yes, you can set your backlog level to include two different rules. The first rule it to set the parent as active when any of its children are set to active. The second rule is to set parent to resolved or closed when all the children are closed. These rules can be configured for backlog level. For more information, see Automate work item state transitions

Why are my work items automatically changing state?

Work items automatically change state after your team administrator set up automation rules on the backlog. Create rules to automatically set the state to active or closed depending on the state change of its children. For more information, see Automate work item state transitions.

Why are the rules not triggering when I reactivate a child item?

Rules only trigger when the state of the child item moves from New to Active or into Closed. Rules don't trigger when the state of the child item moves from Closed back to Active. This is currently by design.

Can I set automation rules per work item type or state?

No, the rules are applied to all work item types in that backlog level. We use state categories instead of states because state categories are consistent across all work item types.

Can I set up automation rules for user stories but not for features or epics?

Yes, rules are configured per backlog level. Automation rules on the user stories backlog don't affect features or epics. But, you can also set up rules separately for features and epics. For more information, see Automate work item state transitions.

Why are automation rules not working if the child or parent items belongs to a different team or project?

Rules only trigger if both the parent and children belong to the same team. We don’t want rules automatically triggered for items that belong to other teams. For more information, see Automate work item state transitions.

Backlogs and boards

What's the difference between a backlog and a board?

Each backlog and board represents a filtered set of work items based on team area path and iteration path assignments. Backlogs list work items, boards display work items as

cards. To understand how the filtering is applied, see About teams and Agile tools, Team defaults referenced by backlogs and boards.

How do I add a backlog or board?

To add a backlog or board, you add a team. Each team is configured with its own set of backlogs and boards as described in About teams and Agile tools. Each team can customize these Agile tools.

What limits should I be aware of?

Azure DevOps imposes limits such as the number of work items that display on a backlog or board, numbers of teams you can define, and more. For a full list, see Work tracking, process, and project limits.

How do I migrate my existing backlog to Azure Boards?

See Azure Boards migration and integration.

How do the three types of backlogs--product, portfolio, and sprint backlogs--differ?

Each backlog lists a filtered set of work items based on the team's selected area path, iteration paths, and work item types.

- **Product backlog**: By default lists User Stories (Agile), Issues (Basic), Product Backlog Items and Bugs (Scrum), or Requirements (CMMI). Provides options to show **Parents**, **Forecast**, and **In Progress** or **Completed** child items.
- **Portfolio backlog**: By default lists Features (all process models) for the Features backlog, and Epics (Agile, Scrum, and CMMI) for the Epic backlog. Provides options to show **Parents** and **In Progress** or **Completed** child items.
- **Sprint backlog**: By default lists all product backlog items assigned to the selected iteration, regardless of status. Provides options to show **Work details**.

Can I define sprints and use them with my board?

Yes. You can assign sprints to work items and filter your board based on the iteration path. For more information, see Filter your board.

If I manage bugs with tasks, can I add bugs as a checklist to a requirement?

No. Task checklists only support the task work item type.

How do I create a view of the critical path?

Azure DevOps doesn't provide a native view of the critical path. In part, as Agile methodologies favor a Minimum Viable Product (MVP) over Critical Path Management (CPM). By using MVP, you identify the shortest path and dependencies by prioritizing epics, features, stories and tasks.

That said, we recommend that you use Delivery Plans to view dependencies and a calendar view of work.

If your organization supports connection to Microsoft Project, you might find more Marketplace extensions that support connection of Azure DevOps to Microsoft Project.

For more context, see The Critical Path on Agile Projects and Running a lean startup on Azure DevOps.

Rollup

How can I get a rollup of Story Points, Effort, or other work item fields?

Rollup columns allow you to view progress bars or totals of numeric fields or descendant items within a hierarchy. Descendant items correspond to all child items within the hierarchy. You can add one or more rollup columns to a product or portfolio backlog. You can add rollup columns to a product or portfolio backlog. See {Display rollup progress or totals](backlogs/display-rollup.md).

Can I get a rollup of team capacity?

No. The data entered for team capacity isn't stored in the regular data stores.

Boards

Is there a way to widen columns on a board?

No. This feature isn't supported. It's a suggested feature, which you can upvote by going to our Developer Community page.

Can I query based on board columns?

Yes, see Query by assignment or workflow changes, board change queries.

Can I view a query as a board?

Yes, by adding the Query Based Boards Marketplace extension.

Is there a way to copy a board configuration to another team?

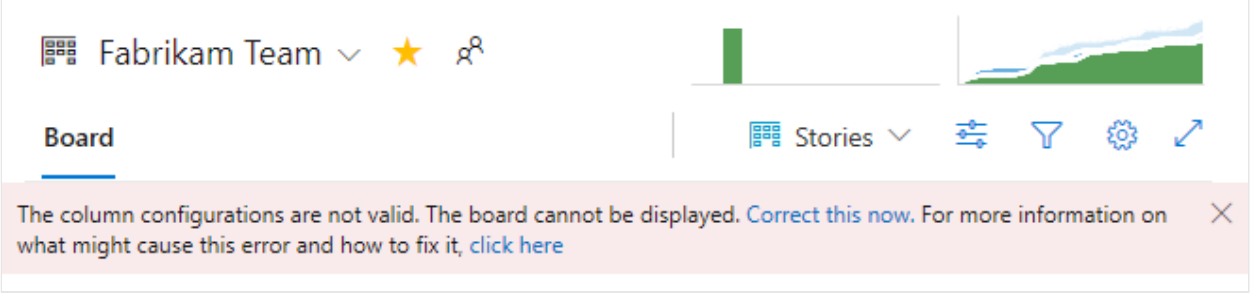
Yes, by adding the Azure Boards Kanban Tools Marketplace extension.

Can I list items based on their column assignment?

Yes, you can track column moves on a board by using the Board Column and Board Column Done fields.

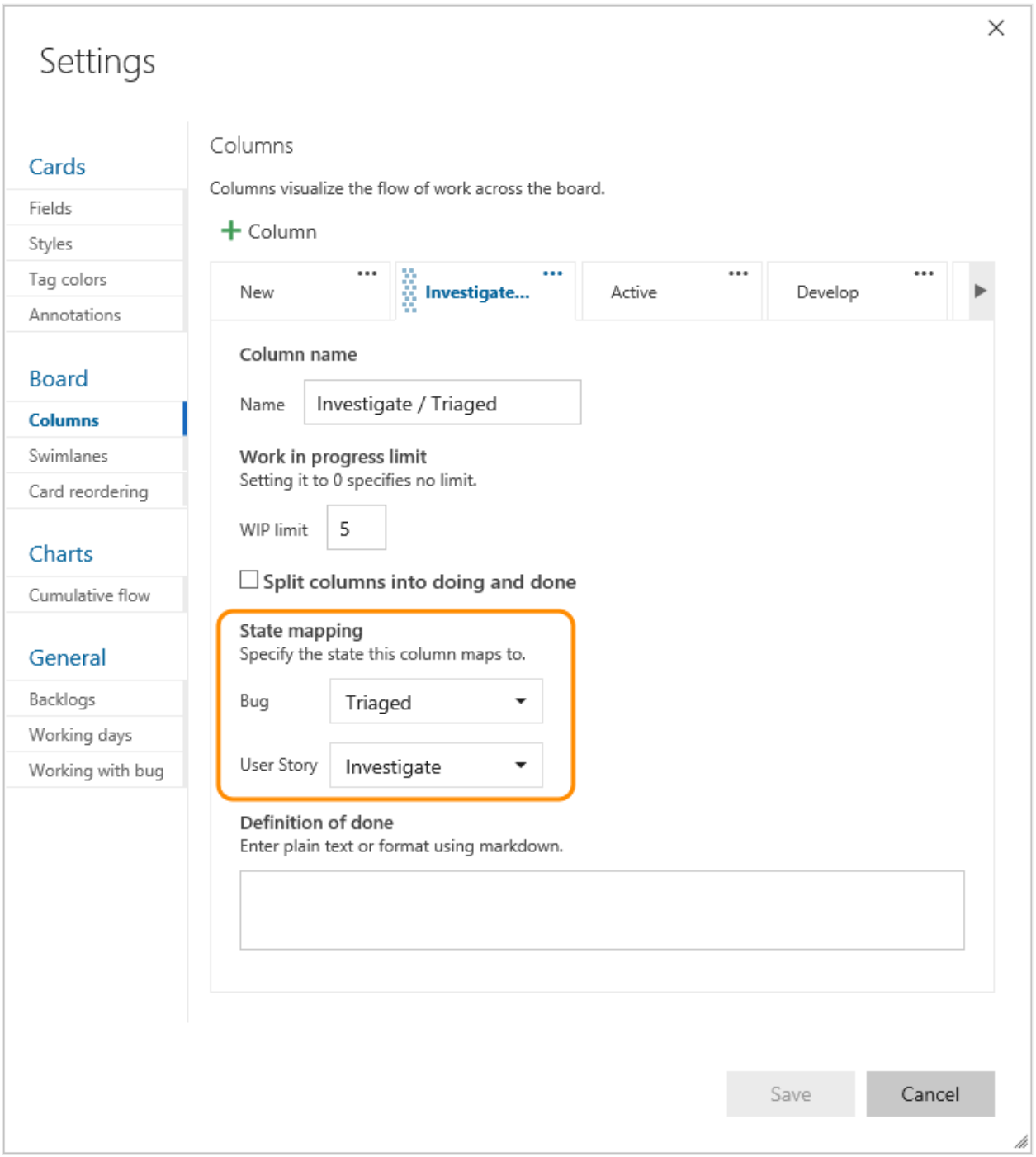
What if I get the error message, "The column configurations aren't valid"?

If you get the following error when you open your board, you need to correct the configuration. The main reason for this error is that the workflow states of work item types added to the **Requirement** category aren't mapped to the column.



Select **Correct this now** to open the **Settings** dialog. In the following example, two new states are added: **Triaged** for bug, and **Investigate** for user story. Each state is then

mapped to an existing or new column. After each state is mapped to a column, the board displays the work items assigned to these states.



Can I use swimlanes and set up swimlane rules?

Yes, you can add or remove swimlanes on your board. You can also set up swimlane rules, where when certain conditions are met, Azure Boards automatically moves work items into specific lanes.

Work item templates

Where should I start to learn about work item templates?

You can define work item templates for teams you belong to. To define work item templates to specify defaults for select fields, see Use templates to add and update work items.

How do I set a default template for a team?

The feature to set a default template for a team isn't a supported feature at this time.

Can I copy a work item template to another team or project?

No. This feature isn't supported at this time.

Can I create a work item template that creates links to other work items?

Example request: *When creating a template, I would like the Parent User Story to be defaulted. There isn't a predetermined field in the template. Would/could this function be under a user-defined selection?*

No, there's no native support for creating hierarchy templates. In particular, you can't specify a default parent work item. You can, however, quickly copy tasks, bugs, and other items using Excel to apply parent-child links in a tree list. Or, you can use a board to add child tasks, backlog items, or features. For more information, see:

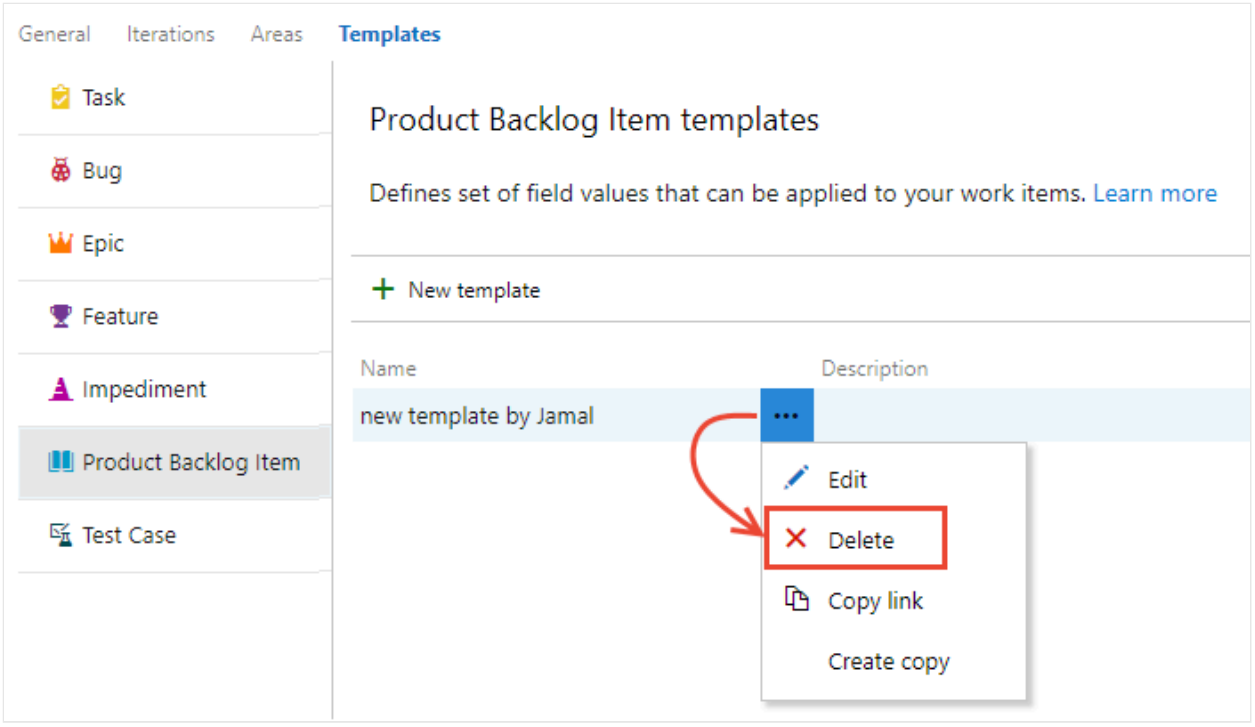
- Add task checklists
- Add, run, and update inline tests
- Add features and epics

Alternatively, you might find a solution for creating child work items by installing one of the following Marketplace extensions:

- Work item form one-click actions
- 1-Click Child-Links
- 1-Click Tasks

How do I delete a work item template?

From the work item type page, choose ··· the actions icon for an existing template and select the **Delete** option.



GitHub integration

How do I connect Azure Boards to GitHub?

Azure Boards integrates with GitHub for Azure DevOps Server 2019 and later versions. For more information, see Azure Boards & GitHub.

Can I specify the status when linking a work item to a GitHub commit or PR?

No. This feature isn't supported at this time.

Configuration and customizations

What is configurable or customizable?

Configuration and customization of Azure Boards occurs at the project and team level. For an overview of what you can configure and customize to support your business needs, see Configuration and customization of Azure Boards.

For FAQs on configuration and customization, see Azure Boards Configuration and Customization FAQs.

Related articles

- Query FAQs
- Work tracking, process, and project limits
- Azure Boards Configuration and Customization FAQs
- FAQs: Work in Excel connected to Azure Boards
- Work across projects FAQs
- Azure Boards extensions
- About teams and Agile tools

Feedback

Was this page helpful? Yes No

Provide product feedback

Azure Boards configuration and customization FAQs

FAQ

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

You have many options to configure and customize an Azure Boards project. If you are new to this subject, see Azure Boards Configuration and Customization. Here, you'll find answers to frequently asked questions when configuring or customizing Azure Boards. For FAQs specific to using Azure Boards or working with queries, see Azure Boards FAQs and Query FAQs.

General

Can sub hubs within Azure Boards be disabled?

No. While you can disable an entire service— such as Azure Boards, Azure Repos, Azure Pipelines— you can't disable individual pages within a service. For more information, see Turn a service on or off.

Are there set limits regarding configuration or customization?

There are a number of limits Azure DevOps imposes such as the number of Area Paths and Iteration Paths you can define for a project, the number of processes you can define, and more. For a full list, see Work tracking, process, and project limits.

Configuration

Who has permissions to configure Azure Boards?

In general, you need to be a member of the Project Administrators group to configure the following project-level objects. Or, you can provide permissions at the node level or object level to individual members or groups. For more information, see Change project-level permissions.

- Area Paths

- Modify work items under an area path
- Iteration Paths
- Shared Queries and Shared Query folders

To grant permissions to an individual member or group to perform the following tasks, you must set the corresponding project-level permissions to **Allow**. See Change project-level permissions.

- Create tag definition
- Delete and restore work items
- Move work items out of a project
- Permanently delete work items

In general, you need to be a member of a team or granted the Team Administrator role to configure the following team-level objects.

- Default and selected Area Paths
- Default and selected Iteration Paths
- Define work item templates

Customization

Who has permissions to customize Azure Boards?

In general, you need to be a member of the Project Collection Administrators group to customize an Inherited process. Or, you must have the following explicit collection-level permissions set to **Allow**:

- Create process
- Delete field from organization
- Delete process
- Edit process

To grant a project member permissions, see Change project-level permissions.

Is there a way to determine who changed what within a process?

Auditing of process changes is available for Azure DevOps Services (cloud) only. For more information, see Access, export, and filter audit logs.

A best practice is to maintain process templates under source control. In that way, changes made to a process is available through standard source control versioning.

How do I change the picklist of values?

Picklist value entries are dependent on the data type. Refer to the following articles based on the field type of interest:

- **Numeric or string picklist fields**: You can customize the menu values for fields such as Priority, Activity, Discipline, and so on. See Add and manage fields (Inherited process) or Add or modify a field, customize a picklist (On-premises XML process).
- **Area Path or Iteration Path**: These tree-path fields are configured for each project. For more information, see About area and iteration (sprint) paths.
- **Person-name or Identity fields**: To add values for fields associated with user accounts such as Assigned To, add users to a project security group or a team. By default, the list for these fields contain the account names for all users and groups that have been added to your organization or on-premises collection. These accounts are often synchronized with Active Directory or Microsoft Entra ID. See Add Active Directory / Microsoft Entra users or groups to a built-in security group.

Can I add or rename a portfolio backlog?

Yes. You can add up to a total of five portfolio backlogs. And, you can rename a portfolio backlog and add custom work item types to a portfolio backlog.

To learn how, see Customize your backlogs or boards (Inheritance process) or Add a portfolio backlog level (On-premises XML process).

How do I add my custom work item type to a backlog?

Modify your requirement backlog to include the custom work item type on the product backlog. For details, see Customize your backlogs or boards, Edit or rename the requirement backlog.

> **Note**
>
> The backlog level to which you add a custom work item type determines the parent work item type(s) for the work item type.

Can caching impact rules applied to work items?

Yes. Conditional rules based on user or group membership are cached for your web browser. If you find yourself restricted to update a work item, you may have encountered one of these rules. If you believe you've encountered an issue that doesn't apply to you, see Work item form IndexDB caching issues. To learn more about conditional rules, see Rules and rule evaluation.

Can I upload my own custom work item type icons for an Inherited process?

Example request: *When I create a new work item type, I get an option to choose an icon. The icon set is limited and doesn't support the icons that we would prefer for our use case. Is there a way to associate custom icon to new work item type?*

No. Uploading your custom icons isn't a supported feature at this time.

Related articles

- Azure Boards Configuration and Customization
- Azure Boards FAQs
- Work tracking, process, and project limits
- Azure Boards extensions
- About teams and Agile tools

Feedback

Was this page helpful? Yes No

Provide product feedback

Set up your project's backlogs and boards in Azure Boards

Article • 11/22/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

In most cases, you can start using your product and portfolio backlogs once your project is created. A default team gets created along with associated backlogs and boards. You can start adding work items to your product backlog using the Backlog or Board.

However, you might need to ensure you've configured your backlogs and boards correctly. Ensure the configuration if you've added a team and want to start using the team backlogs and boards. Changes may be made to a project or team configuration over time. These changes can influence the work items that appear on your backlog and boards.

For an overview of the tools associated with your team, see Manage and configure team tools.

Prerequisites

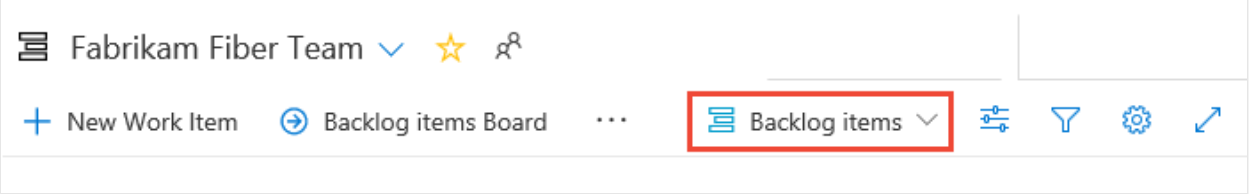
- **Project access**: Be a project member.
- **Permissions**:
  - Be a member of the **Contributors** or **Project Administrators** security group.
  - To view or modify work items, have your **View work items in this node** and **Edit work items in this node** permissions set to **Allow**. By default, the **Contributors** group has this permission set to **Allow**. For more information, see Set work tracking permissions.
- **Access levels**: To add or modify work items, have at least **Basic** access. Users with **Stakeholder** access for public projects have full access to backlog and board features, like users with **Basic** access. For more information, see Stakeholder access quick reference.
- **Defined iterations**: To use the **Planning** pane, ensure your team administrator defined iteration (sprint) paths and configure team iterations.

Default backlog and board work items

The first thing you need to understand is that your product **Backlog** and **Board** display work items that meet the following criteria:

- Work item type belongs to the Requirements category. The types differ depending on the process selected for your project:
  - Basic : Issue, Backlog name=**Issues**
  - Agile: User Story, Backlog name=**Stories**
  - Scrum: Product Backlog Item, Backlog name=**Backlog items**
  - CMMI: Requirement, Backlog name=**Requirements**
- Work item **Area Path** matches one of the selected team's Area Paths
- Work item **Iteration Path** is under the team's Default Iteration Path

You can determine the work item types that belong to your Requirements category. Determine the items by opening your product Backlog and checking the product backlog name.



Look up your team's Area Path(s) and Iteration Paths. For more information, see Define area paths and assign to a team and Define sprint paths and configure team iterations.

Default sprint backlog and Taskboard work items

Your sprint backlog and Taskboard apply the filters associated with your team's default backlog and board work items along with the Iteration Path you select.

You can only select Iteration Paths that have been preselected by your team.

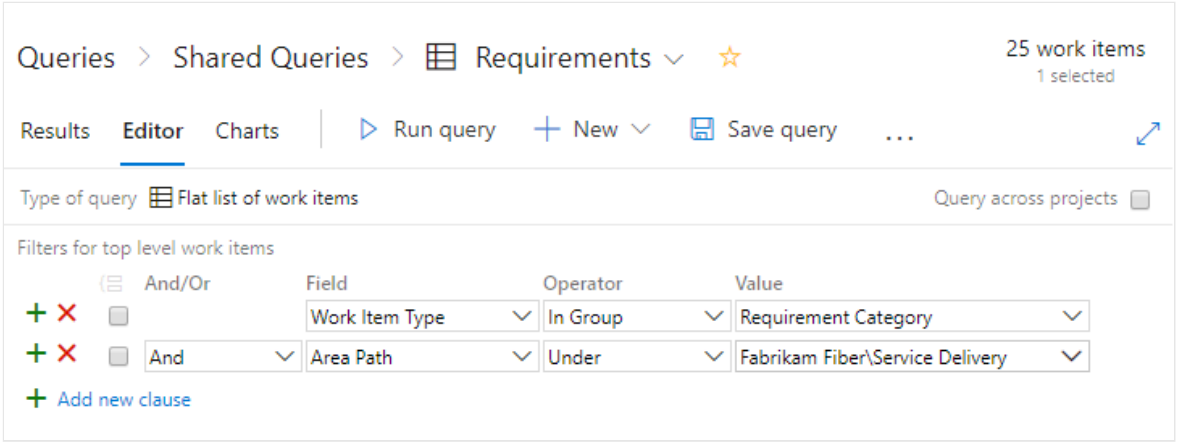
Your sprint backlog displays only those work items assigned to the selected sprint. Child tasks assigned to other sprints aren't displayed.

Review checklist for work items, backlogs, and boards

If you don't see the work items you expect on your product Backlog or board, complete the following checks:

1. Make sure you've selected the team backlog or board of interest. To learn how, see Use breadcrumbs and selectors to go to and open artifacts.

2. Create a query of your backlog items, specifying the work item types that belong to your Requirements category and the Area Path associated with your team, for example:



3. Add the **State**, **Area Path**, and **Iteration Path** fields to the column options.

4. Check the query results and that the values of the work items you expect to show up on your backlog meet these criteria:

   - **Area Path** belongs to your team's area path(s)
   - **Iteration Path** belongs under your team's default iteration path
   - **State** isn't Closed, Completed, Done, or Removed.

> ⓘ Note
>
> You can also filter your product backlog to show or hide work items that are in an **In Progress** state category, corresponding to an Active, Resolved, Committed, Doing workflow state.

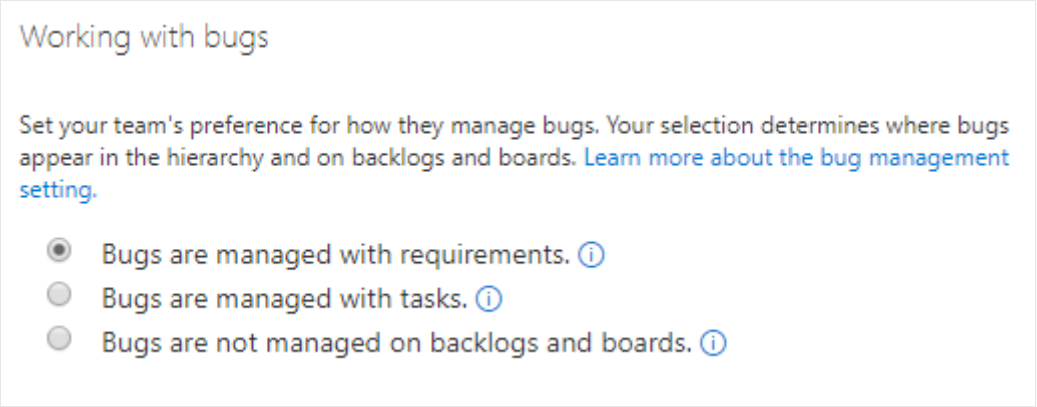
Add bugs to your backlogs and boards

For all processes except the Basic process, each team manages the way bugs are tracked. Track bugs in the Requirements category because they show up on the Backlog and board or the Tasks category. They can also show up on the Taskboard or the Bugs category where they don't appear on either backlogs or boards.

> ⓘ Note

Bug work item types aren't available with the Basic process. The Basic process tracks bugs as Issues and is available when you create a new project from Azure DevOps Services or Azure DevOps Server 2019.1 or later versions.

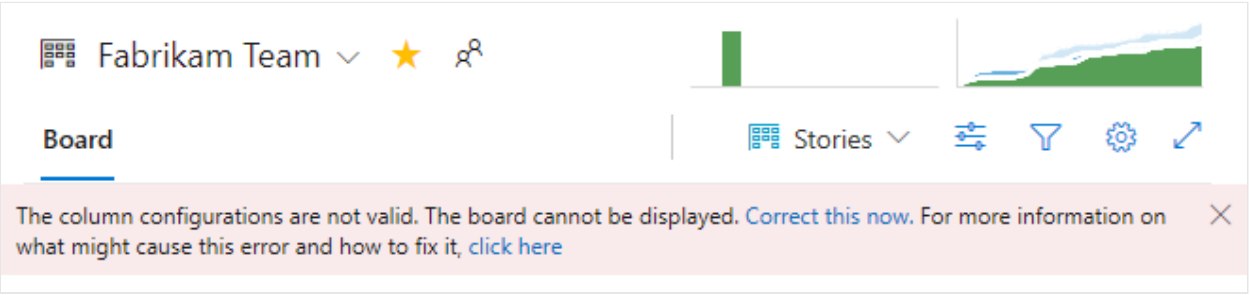
If you want bugs to show up on your Backlog and Board, choose **Bugs are managed with requirements**.



For more information, see Show bugs on backlogs and boards.

Correct your board configuration

If you see the following error when you open your board, you need to correct the configuration. The main reason for this error is that the workflow states of work item types that have been added to the Requirements category aren't mapped to the column.



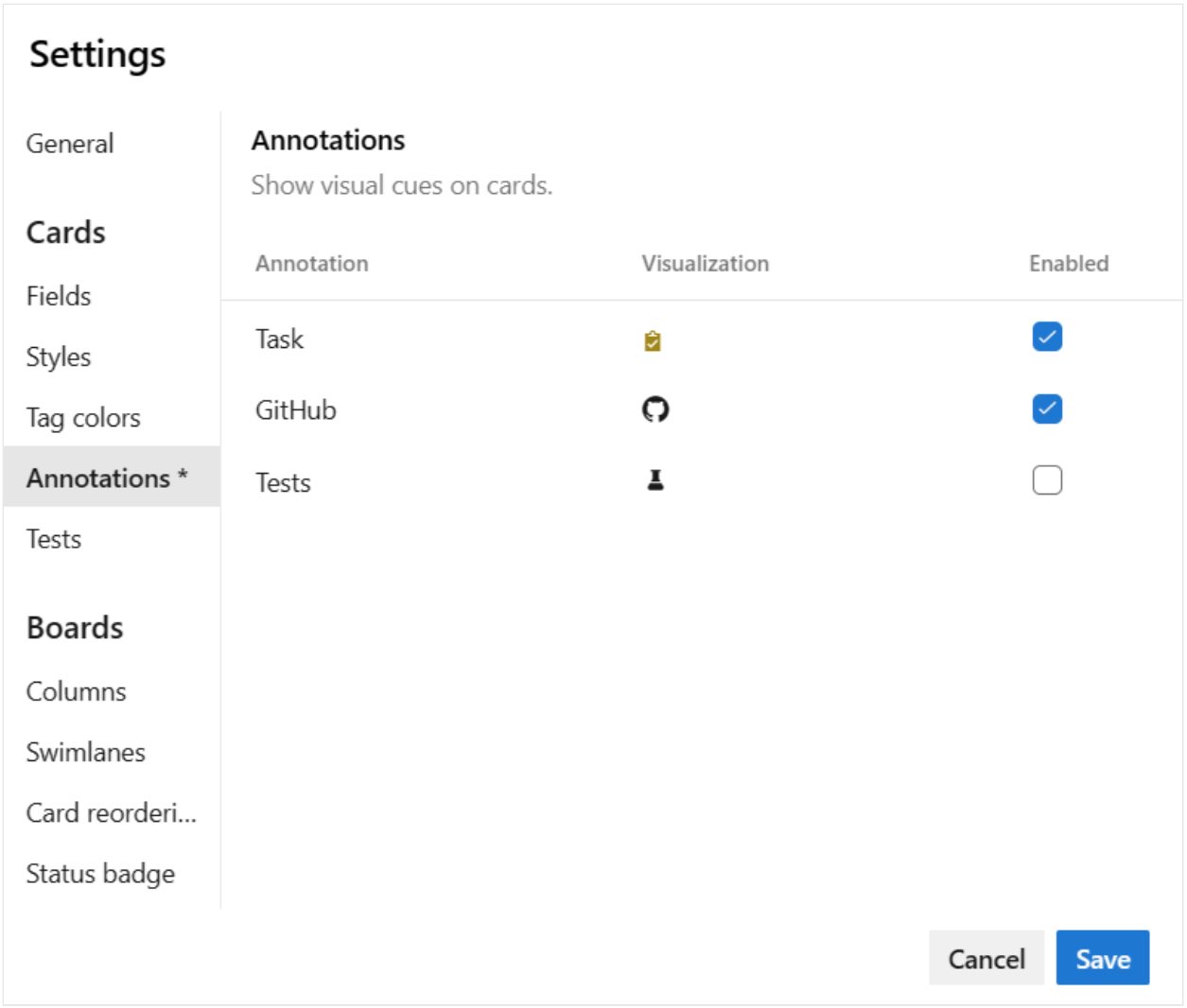
Choose **Correct this now** to open the Settings dialog. To map the workflow states, refer to Add columns to your board, Update column-to-State mappings.

Customize your board checklist items

Checklists are a great way to create work items that are automatically linked with a parent-child link to another work item on a board. You can customize the work item types that you can add as a checklist by opening the Board Settings, choose

Annotations, and enable the work item types you want to appear on the board. For more information, see Customize cards.

For example, here we've chosen to track bugs along with tasks, and enable Task, Bug, GitHub objects, and Tests to appear within checklists.



For more information about checklists, see the following articles:

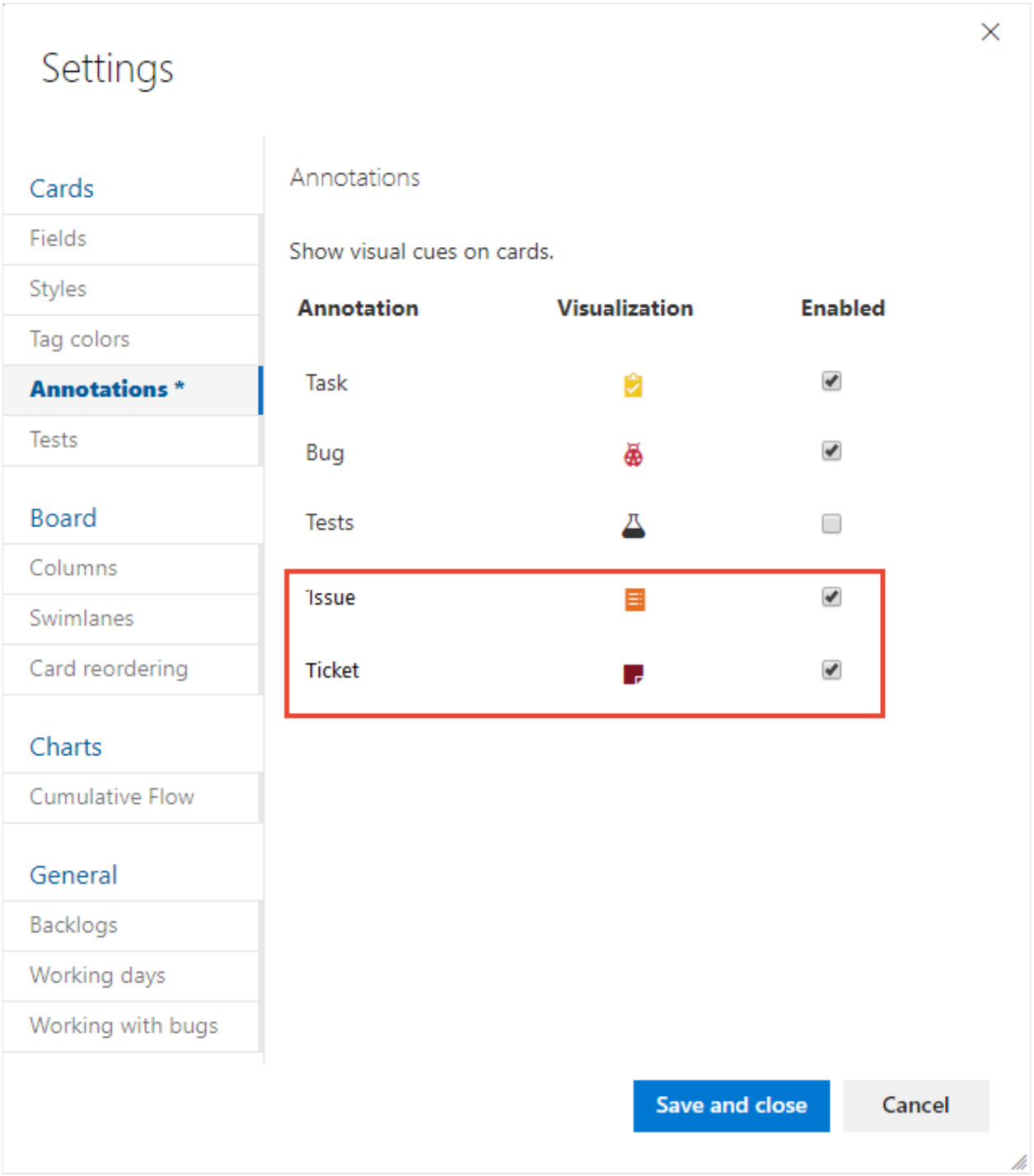
- Add tasks or child items as checklists
- Add, run, and update inline tests
- Link GitHub commits, pull requests, branches, and issues to work items

Add other work item types to your board checklist

If you added work item types to the Task Category as described in Add custom work item types to your Taskboard later in this article, you can choose if these types appear within a checklist on your product board. You make this choice by opening **Board Settings**, choose **Annotations**, and enable the work item types you want to appear on

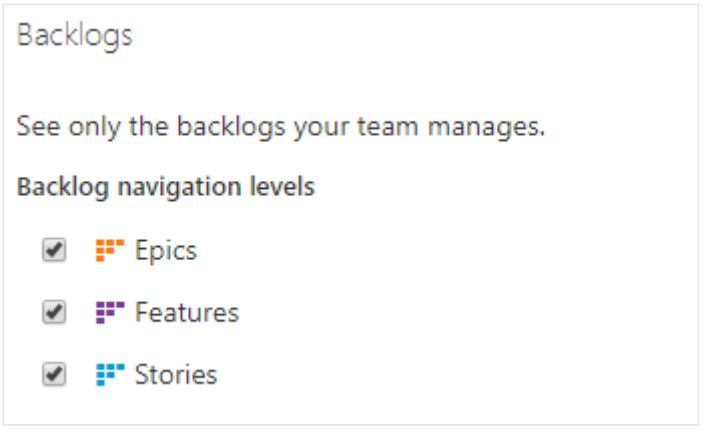
the board. You can enable up to five annotations. For more information, see Customize cards.

For example, here we've chosen to track bugs along with tasks, and we enable Issue and Ticket and Task and Bug. For more information about checklists, see Add tasks or child items as checklists and Add, run, and update inline tests.



Hide or show backlog levels

Your team can also choose to hide or show one or more backlog levels. Feature teams often manage backlog items, while management teams manage features and epics. In this situation, you can enable or disable a backlog level.



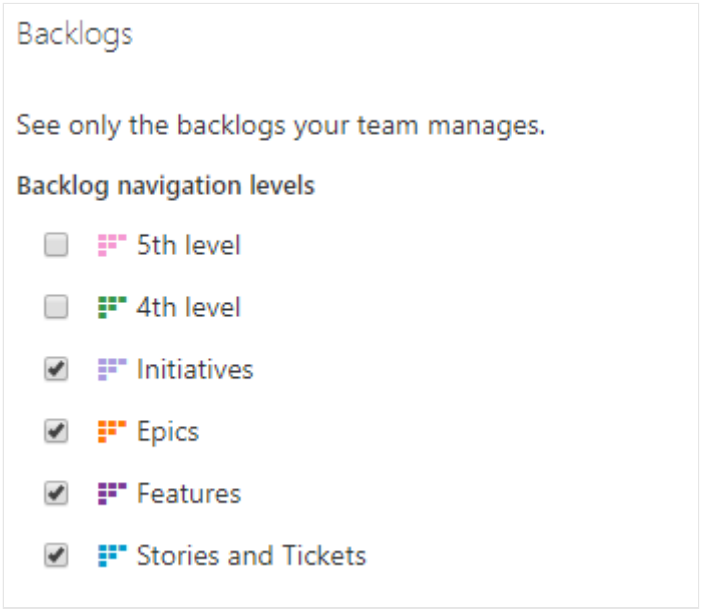
For more information, see Select backlog navigation levels for your team.

Add custom work item types to your backlogs and portfolio backlog levels

If you want to track different work item types on your product backlog, you can do that by adding custom work item types and adding them to a specific backlog level.

You can also add custom work item types and add them to portfolio backlogs. You can add up to five portfolio backlogs.

For example, here we've added Initiatives, fourth level, and fifth level work item types to support five levels of portfolio backlogs. We've also added a custom work item type named Ticket and added that to the product backlog.



For more information, see the following resources:

- Add and manage work item types (Inherited process)
- Customize your backlogs or boards (Inherited process)

- Customize an inheritance process

Add custom work item types to your Taskboard

To add custom work item types to appear on your sprint Taskboard, follow the steps outlined next based on the process model your project uses.

> ⓘ **Note**
>
> You can enable work item types that you add to your Iteration backlog to appear as a checklist on your product board. To learn how, see **Customize your board checklist items** provided earlier in this article.

Track custom work items with the Inherited process model

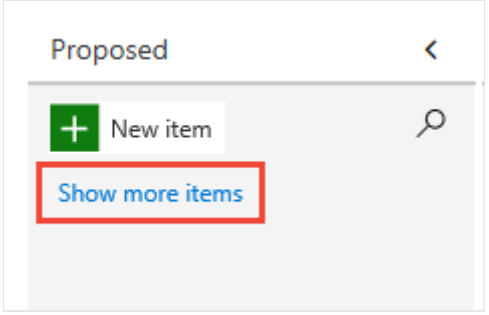
For example, if you want to track a custom work item type, Tickets, along with Tasks and Bugs, you do the following tasks:

1. Define the Ticket custom work item type. See Add and manage work item types.
2. Add the Ticket work item types to the Iteration backlog. For more information, see Customize your backlogs or boards for a process.

Other factors that can affect work items in your backlogs and boards

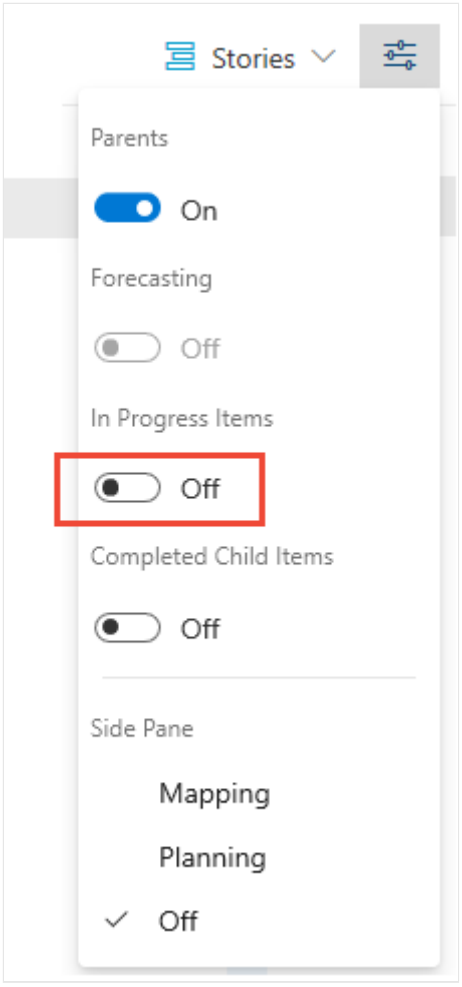
The following settings can influence the type and number of work items that appear in your backlogs and boards.

- In your board, newly added work items may not appear if they're stack ranked lower within the product backlog. By choosing **Show more items**, you can cause the board to refresh and display more work items.



- If you have nested work items that belong to the same category, only leaf nodes may appear on the board (for TFS 2018.1 and earlier versions). For this reason, we recommend that you don't nest work items of the same work item type or belonging to the same category. For more information, see Fix reordering and nesting issues, How backlogs and boards display hierarchical (nested) items.

- If you've turned off the **In Progress** view, then those work items where work has started won't appear in the backlog list.



- Work items appear in the priority order in which they're added or moved to. This order or sequence is managed by the **Stack Rank** (Basic, Agile, and CMMI processes) or **Backlog Priority** (Scrum) field. For more information, see the Stack rank section in Backlogs, portfolios, and Agile project management.

- Each backlog can display up to 999 work items. If your backlog exceeds this limit, then you may want to consider adding a team and moving some of the work items to the other team's backlog.

- Sprint backlogs show only those work items that meet the team's area path and the **Iteration Path** defined for the sprint.

- Inheritance process model: If an administrator disables or deletes a work item type, it doesn't appear on backlogs and boards.
- On-premises XML process model: If an administrator deletes or destroys a work item type, it doesn't appear on backlogs and boards.

Related articles

- Add a team, move from one default team to several teams
- Create your backlog
- Backlog priority or stack rank order
- Use categories to group work item types
- Workflow states & state categories
- Automate work item state transitions

Feedback

Was this page helpful? Yes No

Provide product feedback

Troubleshoot reordering and nesting issues

Article • 10/14/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

When you reorder, nest, and display work items, Azure Boards expects a natural hierarchy. The natural hierarchy breaks when you create same-category or same-type links between work items. For example, parent to child links that are bug to bug or user story to user story or *requirements* category to *task* category. Use this article to address error messages when you add links that aren't in the natural hierarchy.

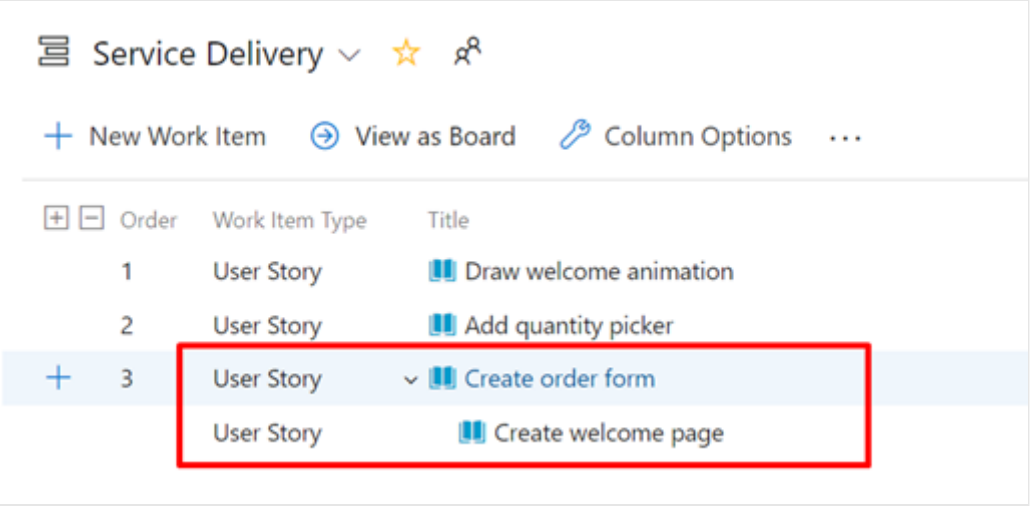
You cannot reorder work items and some work items might not be shown

You might see an error similar to one of the following messages:

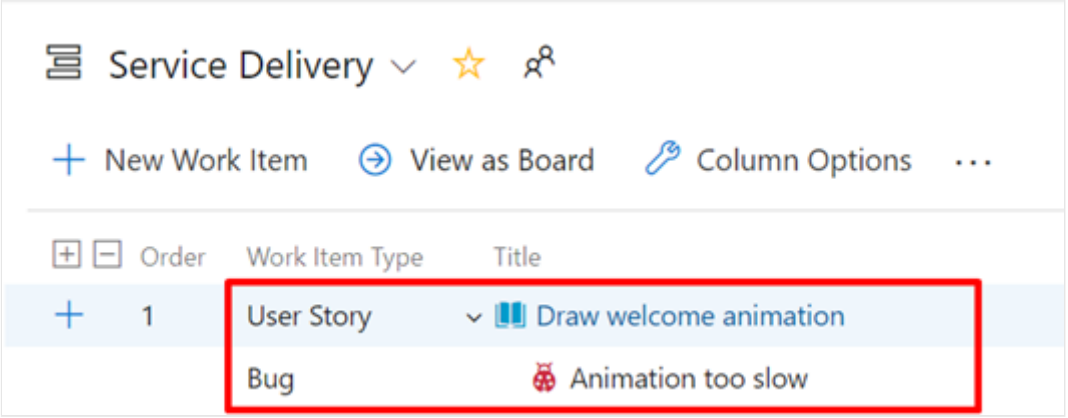
> - You cannot reorder work items and some work items might not be shown
> - No work item IDs are listed

To address this error, do the following steps:

1. Open your backlog.
2. Review the list of items to identify those items of the same type that are nested.
    - **Example #1:** The following image shows a user story as a child of another user story.



- **Example #2:** The following image shows a bug as a child of a user story. When the backlog displays user stories and bugs at the same level (Requirements category), it results in a nested item that disables the ordering feature.



3. Remove any parent-child links that exist among nested items of the same work item type or category, or consider changing the link type to *Related*.
4. Refresh your backlog.

These steps should resolve the issue, and the error message no longer displays.

Work item can't be reordered because its parent is on the same category

You might see an error similar to one of the following messages:

> - You cannot reorder work items and some work items might not be shown. See work item(s) 7 to either remove the parent to child link or change the link type to Related.
> - Work item 3 can't be reordered because its parent is on the same category.

To address this error, do the following steps:

1. Open the work item listed in the error message.
2. Look for a parent or child link. Make sure this link goes to a work item in the same category as the work item you opened. Look for a link that goes to another work item that appears on the same backlog level as the work item you opened. Depending on your team's bug behavior setting, bugs might appear with requirements or tasks.
3. Remove the problem parent-child link. If you would like to keep these items associated, use *Related* link type instead.

The message no longer displays.

Work items in progress might disappear on refresh

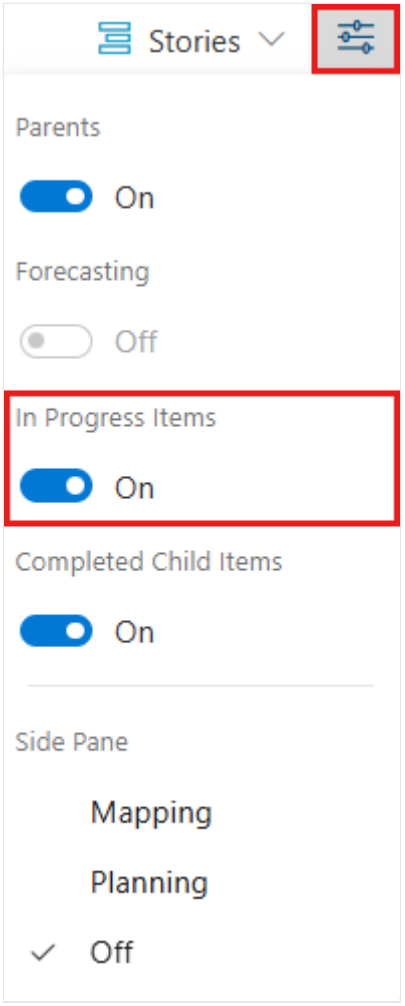
You might see an error similar to the following message:

> Items added to the backlog might disappear on a refresh because your team project marks them as "in progress". Those items appear when you change the "In progress" filter to Show.

This message indicates that the **In Progress** filter for the backlog is turned off.

When you refresh your browser, the work items display based on your selected filters. To reset the filters, do the following steps.

1. Open your backlog.
2. From the **View options** selector, choose to show or hide **In Progress items**.



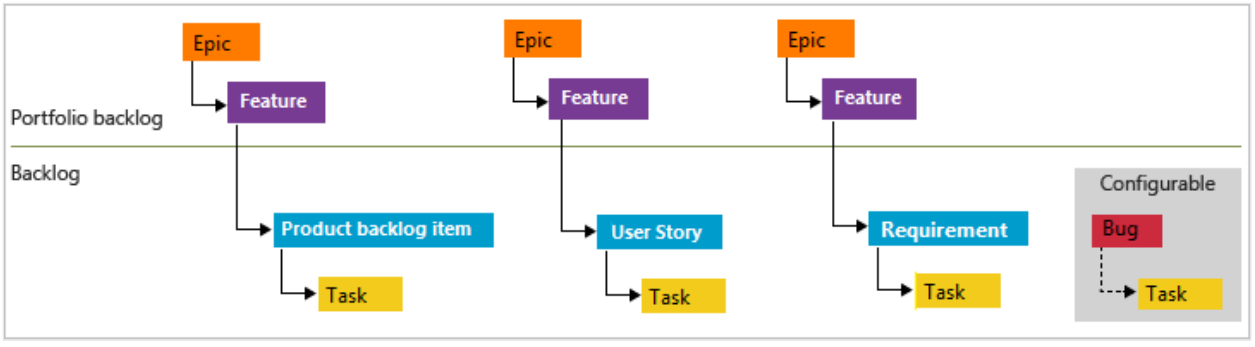
If you turn the **In Progress** control off, then items that are in the *Active*, *Committed*, or *Resolved* states or states that map to the **In Progress** category state don't appear.

Hide **In Progress items** when you want to forecast work. For more information, see Forecast your product backlog.

> ⓘ Note
>
> - For more information, see Configure your backlog view and Add custom work item types.
> - For issues that might occur with multi-team ownership, see Exercising select features with shared area paths.
> - To reorder work items on your backlog, you must have Basic or higher level access. If you have Stakeholder access, you can't reorder work items. For more information, see Stakeholder access quick reference.

Natural hierarchy for work item types

The following image shows the natural hierarchy for the Agile, Scrum, and Capability Maturity Model Integration (CMMI) processes.



Best practices

Do:

- Maintain a flat list, rather than nesting requirements, bugs, and tasks.
- Only create parent-child links one level deep between items that belong to a different category. The category a work item belongs to gets determined by your process levels and your team's selected bug behavior.
- Use the *feature* work item type to group user stories (Agile), issues (Basic), work items (Scrum), or requirements (CMMI). You can map work items to features. This

mapping creates parent-child links in the background. For more information, see Organize your backlog.

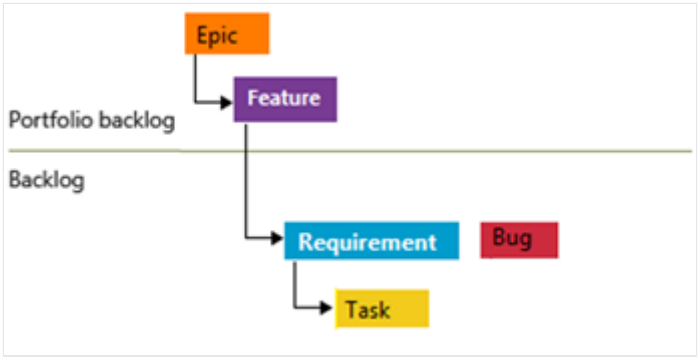
Don't:

- Create a hierarchy of work items, tasks, and bugs.
- Establish same-category hierarchies, such as parent-child links among work items of the same type. For example, create story-story, bug-bug, task-task, or issue-issue links. The backlog, board, and sprints experiences don't support reordering for same-category hierarchies, because that approach introduces confusion by ordering a work item that doesn't belong at that level.

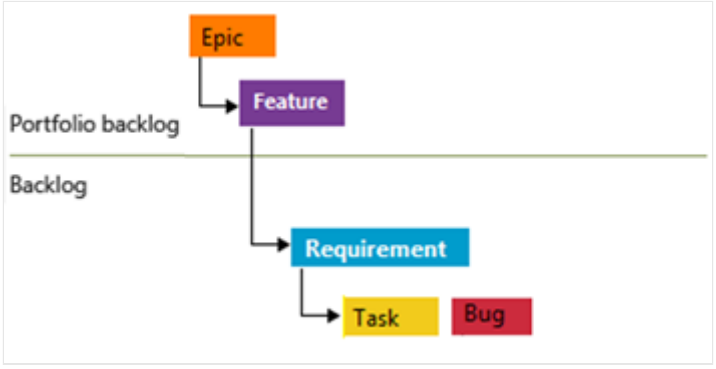
Track bugs as requirements or tasks

Each team has the flexibility to choose to track bugs as requirements, tasks, or neither. See the following guidelines:

- **If you track bugs as *requirements*:** Nest them only under the *Feature* level.



- **If you track bugs as *tasks*:** Nest them only under the *Requirement* level.



For more information, see Show bugs on backlogs and boards.

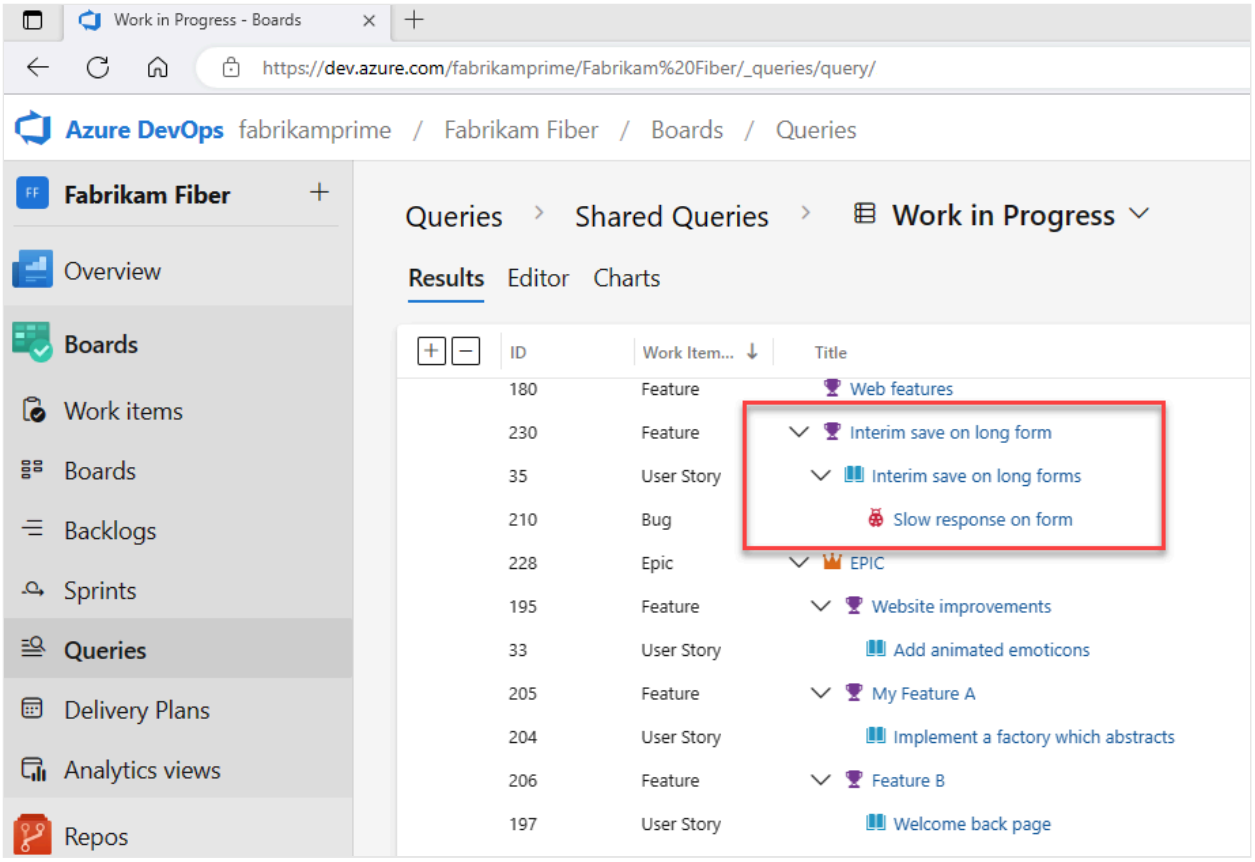
Display nested items on backlogs and boards

Sprint backlogs and Taskboards exclusively display the last node in a same-category hierarchy, which is referred to as the leaf node.

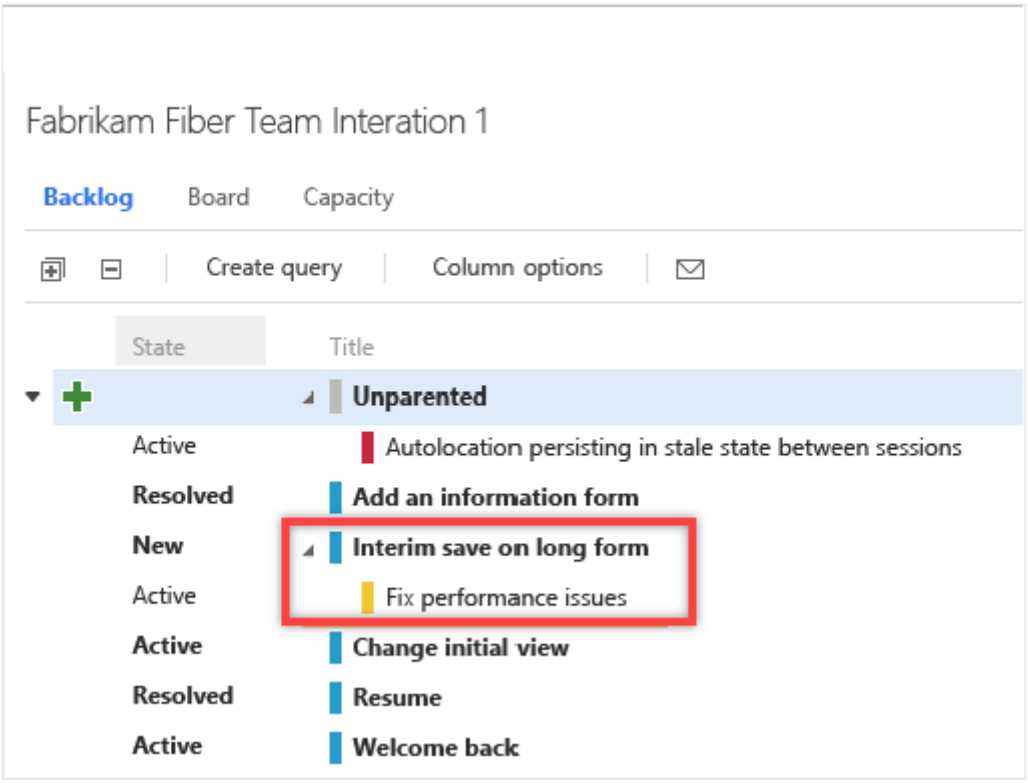
Sprint backlogs and Taskboards

When tasks and bugs link to their parent requirements, they group them correctly on the sprint backlog and Taskboard. When you establish parent-child links between a requirement and a bug, and between the bug and a task, as demonstrated here, the task appears on the sprint backlog and Taskboard, while the bug doesn't.

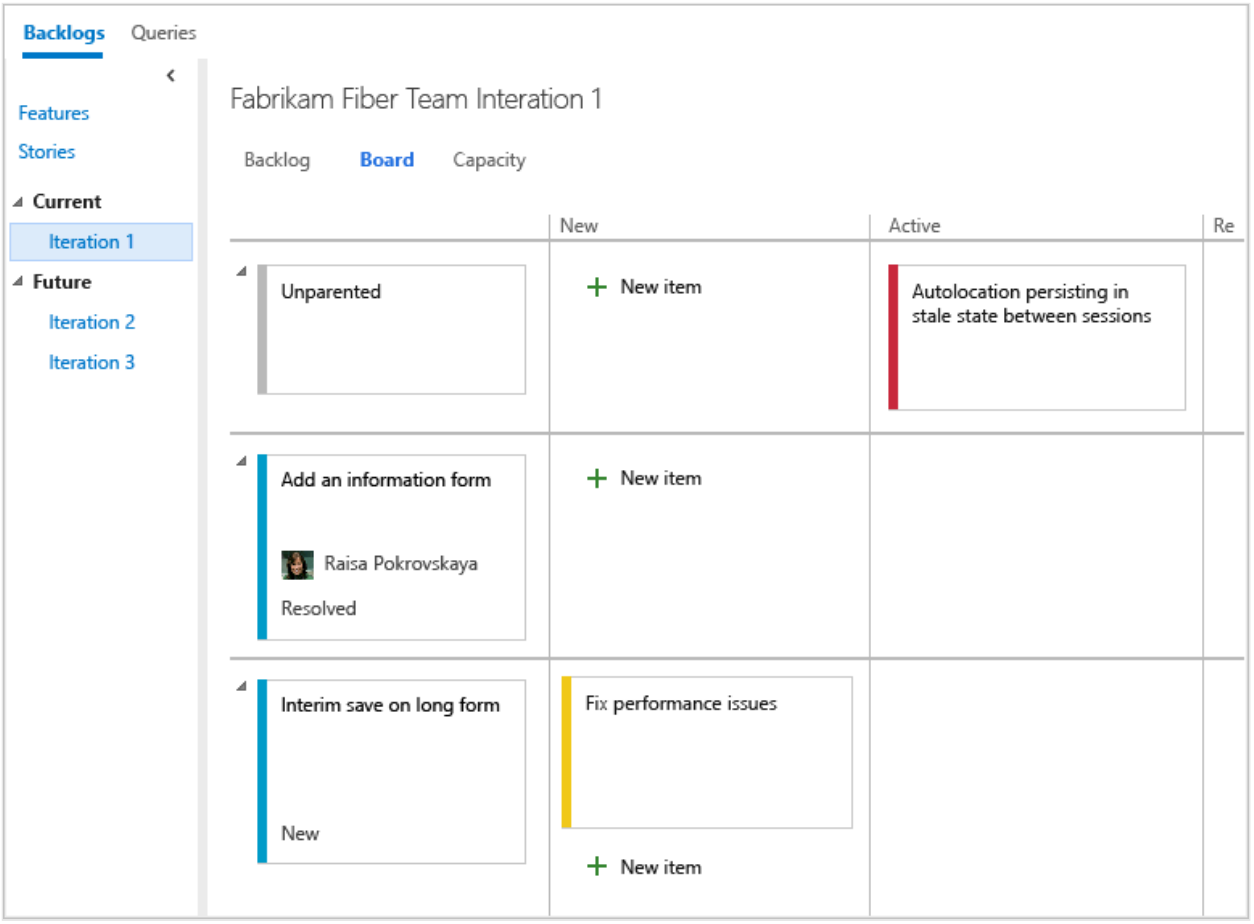
Hierarchy of items assigned to a sprint backlog



Only leaf nodes appear on sprint backlogs



Only leaf nodes appear on Taskboards



Frequently asked questions (FAQs)

Q: Is there a workaround to display intermediate nodes within a hierarchy?

A: No, not at this time. You can always check the entire list of items assigned to a sprint when you select **Create query**.

Related articles

- Set up your backlogs and boards
- Learn about Kanban boards

Feedback

Was this page helpful? Yes No

Provide product feedback

Azure Boards FAQs

FAQ

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Find answers to frequently asked questions about using Azure Boards. For FAQs specific to queries or Microsoft Excel integration to add or modify work items defined in Azure DevOps, see Query FAQs and FAQs: Work in Excel connected to Azure Boards.

You can view a list of features that are on our roadmap for Azure Boards from the Features Timeline. To request a feature or up-vote a feature, go to our Developer Community page.

Access and permissions

Who can contribute to Azure Boards?

As a member of an Azure Boards project, you can use most features to track work. Limitations to select features are based on the access level and security group to which a user is assigned. Other limitations can be imposed based on work tracking permissions or custom rules. For more information, see Default permissions and access for Azure Boards.

How can noncontributors view or modify work items?

For private projects, you can grant access to an unlimited number of users by assigning them Stakeholder access. For public projects, anonymous users—ones who don't sign into the project—can view all work items. For more information, see Stakeholder access quick reference and Make a private project public.
Also, if your organization uses Microsoft Entra ID to manage user access, you can add external users to your organization.

How can I restrict viewing or modifying select work items?

You can restrict access by setting permissions for an area path. For more information, see the following articles:

- Set work tracking and plan permissions
- Grant or restrict permissions to select tasks

Can caching affect rules applied to work items?

Yes. Conditional rules based on user or group membership get cached for your web browser. If you find yourself restricted to update a work item, you probably encountered one of these rules. For more information about conditional rules, see Rules and rule evaluation. If your cache stores outdated rules, you can wait for the client cache to expire within three days, or you can clear the cache.
To clear the cache, run the following command in the browser command window and then refresh the browser: window.indexedDB.deleteDatabase("wit")

What client tools support work tracking with Azure Boards?

See Tools and clients that connect to Azure DevOps.

Work items

Where should I start to learn about work items and work item types?

For an introduction to work items, see About work items.

How do I view all work items?

To view all work items defined in a project, open a query and add two clauses:

- `Work Item Type = [Any]`
- `State = [Any]`

From there, you can add filters to refine the query. For more information, see Create and save managed queries with the query editor.

Can a work item be assigned to several users or a user group?

No. Work items can only be assigned to a single user. Also, you can only add a user that is available from the people picker.

What's the recommended method to group work?

The main method to group work is to assign it to the same area path. Area paths are used to group work items by product, feature, or business areas and to support teams responsible for work assigned to those areas. In addition, you can group work under a parent work item using parent-child links, referred to as a hierarchical grouping.

For a discussion of these two usages and the tools they support, see Configuration and customization of Azure Boards, Area paths, product teams, and portfolio management.

How do I mark a task or work item as a milestone task?

Milestone markers aren't used in Azure Boards work tracking, except for Delivery Plans. Delivery Plans provide a calendar view and allow you to define a milestone marker. For more information, see Review team Delivery Plans.

However, you can use one or more of the following options to mark a work item as a milestone:

- Prepend or append the word **Milestone** in the title of your work item
- Add a work item tag labeled **Milestone**
- Add a custom field labeled **Milestone** and populate it with a pick list of milestones
- Link work items using the Predecessor/Successor or Related link type to a milestone work item
- Assign the milestone work item to the sprint in which it's targeted for completion.

How can I best track dependencies?

You can track dependencies between work items by linking them using a related or other link type. See Link work items to other objects.

You can view dependencies using Delivery Plans.

If you're tracking dependencies across one or more organizations, you might want to use the Dependency Tracker.

What determines which work item type should be a parent for another work item type?

Each process defines four default backlog levels: the iteration backlog, requirement backlog, feature and epic portfolio backlogs. The work item types added to each backlog level are the natural work item types to form parent-child relationship and group work items into a hierarchy. For more information, see Organize your backlog, map child work items to parents. To customize the process, see Customize your backlogs or boards, Edit or rename the requirement backlog.

How do I copy or clone a work item with all linked items?

With Azure Boards (cloud service), you can choose to copy child work items when copying a work item. For more information, see Copy or clone work items.

How do I bulk modify a rich-text field?

See Bulk modify work items, Bulk modify rich-text fields.

Automation rules

Is there way to make the parent active when a child is active?

Yes, you can set your backlog level to include two different rules. The first rule it to set the parent as active when any of its children are set to active. The second rule is to set parent to resolved or closed when all the children are closed. These rules can be configured for backlog level. For more information, see Automate work item state transitions

Why are my work items automatically changing state?

Work items automatically change state after your team administrator set up automation rules on the backlog. Create rules to automatically set the state to active or closed depending on the state change of its children. For more information, see Automate work item state transitions.

Why are the rules not triggering when I reactivate a child item?

Rules only trigger when the state of the child item moves from New to Active or into Closed. Rules don't trigger when the state of the child item moves from Closed back to Active. This is currently by design.

Can I set automation rules per work item type or state?

No, the rules are applied to all work item types in that backlog level. We use state categories instead of states because state categories are consistent across all work item types.

Can I set up automation rules for user stories but not for features or epics?

Yes, rules are configured per backlog level. Automation rules on the user stories backlog don't affect features or epics. But, you can also set up rules separately for features and epics. For more information, see Automate work item state transitions.

Why are automation rules not working if the child or parent items belongs to a different team or project?

Rules only trigger if both the parent and children belong to the same team. We don’t want rules automatically triggered for items that belong to other teams. For more information, see Automate work item state transitions.

Backlogs and boards

What's the difference between a backlog and a board?

Each backlog and board represents a filtered set of work items based on team area path and iteration path assignments. Backlogs list work items, boards display work items as

cards. To understand how the filtering is applied, see About teams and Agile tools, Team defaults referenced by backlogs and boards.

How do I add a backlog or board?

To add a backlog or board, you add a team. Each team is configured with its own set of backlogs and boards as described in About teams and Agile tools. Each team can customize these Agile tools.

What limits should I be aware of?

Azure DevOps imposes limits such as the number of work items that display on a backlog or board, numbers of teams you can define, and more. For a full list, see Work tracking, process, and project limits.

How do I migrate my existing backlog to Azure Boards?

See Azure Boards migration and integration.

How do the three types of backlogs--product, portfolio, and sprint backlogs--differ?

Each backlog lists a filtered set of work items based on the team's selected area path, iteration paths, and work item types.

- **Product backlog**: By default lists User Stories (Agile), Issues (Basic), Product Backlog Items and Bugs (Scrum), or Requirements (CMMI). Provides options to show **Parents**, **Forecast**, and **In Progress** or **Completed** child items.
- **Portfolio backlog**: By default lists Features (all process models) for the Features backlog, and Epics (Agile, Scrum, and CMMI) for the Epic backlog. Provides options to show **Parents** and **In Progress** or **Completed** child items.
- **Sprint backlog**: By default lists all product backlog items assigned to the selected iteration, regardless of status. Provides options to show **Work details**.

Can I define sprints and use them with my board?

Yes. You can assign sprints to work items and filter your board based on the iteration path. For more information, see Filter your board.

If I manage bugs with tasks, can I add bugs as a checklist to a requirement?

No. Task checklists only support the task work item type.

How do I create a view of the critical path?

Azure DevOps doesn't provide a native view of the critical path. In part, as Agile methodologies favor a Minimum Viable Product (MVP) over Critical Path Management (CPM). By using MVP, you identify the shortest path and dependencies by prioritizing epics, features, stories and tasks.

That said, we recommend that you use Delivery Plans to view dependencies and a calendar view of work.

If your organization supports connection to Microsoft Project, you might find more Marketplace extensions that support connection of Azure DevOps to Microsoft Project.

For more context, see The Critical Path on Agile Projects and Running a lean startup on Azure DevOps.

Rollup

How can I get a rollup of Story Points, Effort, or other work item fields?

Rollup columns allow you to view progress bars or totals of numeric fields or descendant items within a hierarchy. Descendant items correspond to all child items within the hierarchy. You can add one or more rollup columns to a product or portfolio backlog. You can add rollup columns to a product or portfolio backlog. See {Display rollup progress or totals](backlogs/display-rollup.md).

Can I get a rollup of team capacity?

No. The data entered for team capacity isn't stored in the regular data stores.

Boards

Is there a way to widen columns on a board?

No. This feature isn't supported. It's a suggested feature, which you can upvote by going to our Developer Community page.

Can I query based on board columns?

Yes, see Query by assignment or workflow changes, board change queries.

Can I view a query as a board?

Yes, by adding the Query Based Boards Marketplace extension.

Is there a way to copy a board configuration to another team?

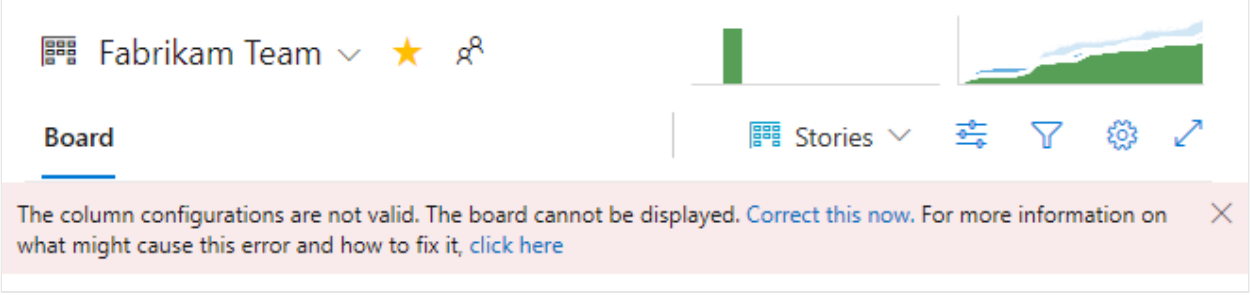
Yes, by adding the Azure Boards Kanban Tools Marketplace extension.

Can I list items based on their column assignment?

Yes, you can track column moves on a board by using the Board Column and Board Column Done fields.

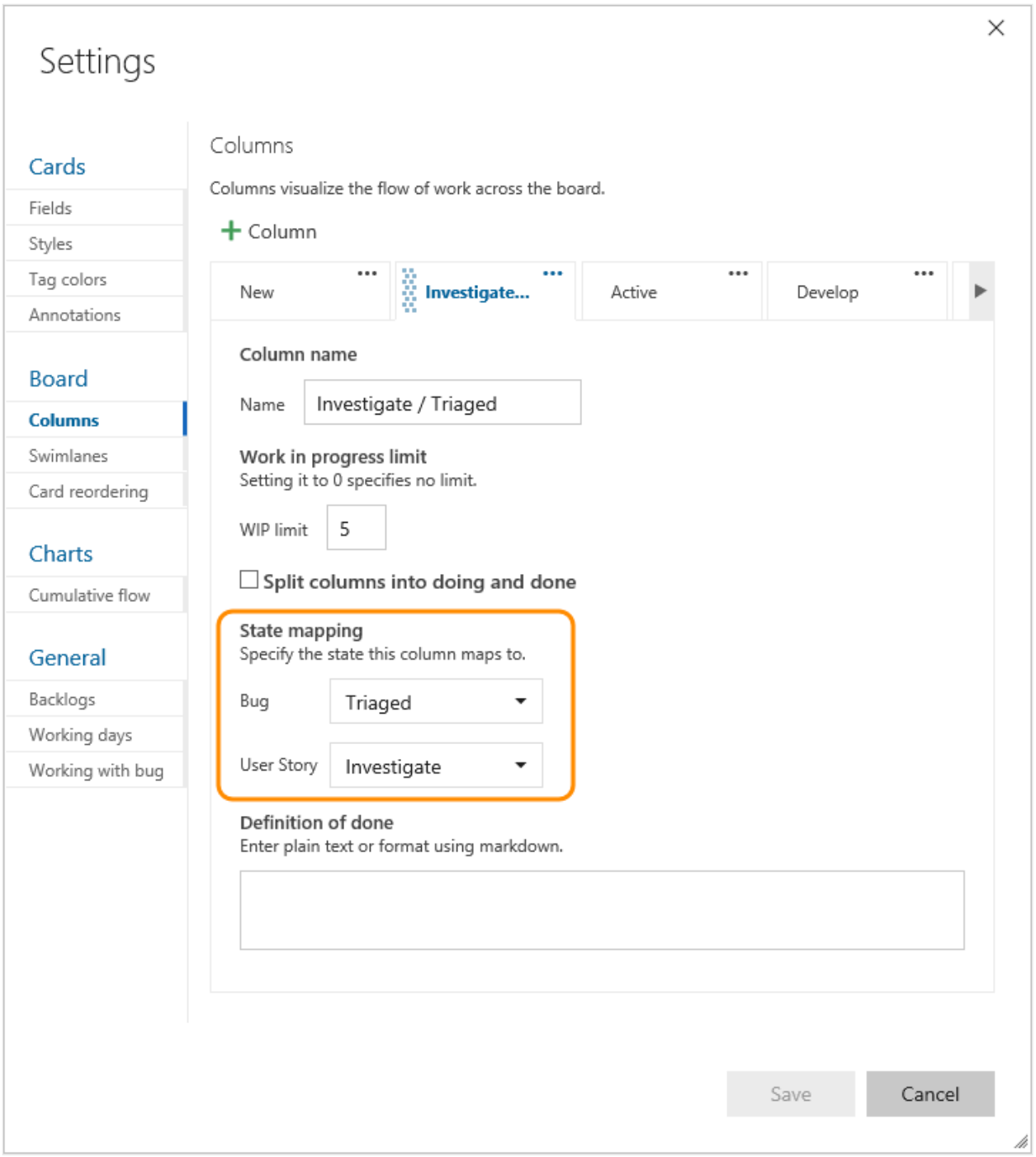
What if I get the error message, "The column configurations aren't valid"?

If you get the following error when you open your board, you need to correct the configuration. The main reason for this error is that the workflow states of work item types added to the **Requirement** category aren't mapped to the column.



Select **Correct this now** to open the **Settings** dialog. In the following example, two new states are added: **Triaged** for bug, and **Investigate** for user story. Each state is then

mapped to an existing or new column. After each state is mapped to a column, the board displays the work items assigned to these states.



Can I use swimlanes and set up swimlane rules?

Yes, you can add or remove swimlanes on your board. You can also set up swimlane rules, where when certain conditions are met, Azure Boards automatically moves work items into specific lanes.

Work item templates

Where should I start to learn about work item templates?

You can define work item templates for teams you belong to. To define work item templates to specify defaults for select fields, see Use templates to add and update work items.

How do I set a default template for a team?

The feature to set a default template for a team isn't a supported feature at this time.

Can I copy a work item template to another team or project?

No. This feature isn't supported at this time.

Can I create a work item template that creates links to other work items?

Example request: *When creating a template, I would like the Parent User Story to be defaulted. There isn't a predetermined field in the template. Would/could this function be under a user-defined selection?*

No, there's no native support for creating hierarchy templates. In particular, you can't specify a default parent work item. You can, however, quickly copy tasks, bugs, and other items using Excel to apply parent-child links in a tree list. Or, you can use a board to add child tasks, backlog items, or features. For more information, see:

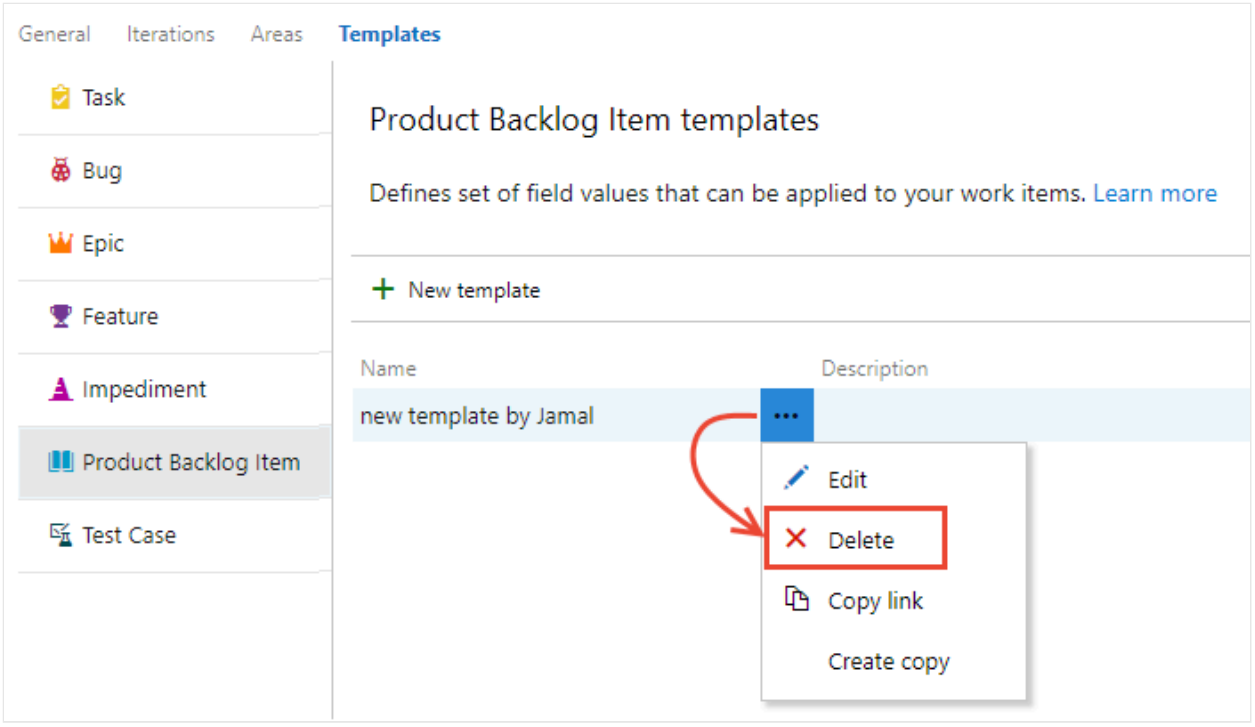
- Add task checklists
- Add, run, and update inline tests
- Add features and epics

Alternatively, you might find a solution for creating child work items by installing one of the following Marketplace extensions:

- Work item form one-click actions
- 1-Click Child-Links
- 1-Click Tasks

How do I delete a work item template?

From the work item type page, choose ··· the actions icon for an existing template and select the **Delete** option.



GitHub integration

How do I connect Azure Boards to GitHub?

Azure Boards integrates with GitHub for Azure DevOps Server 2019 and later versions. For more information, see Azure Boards & GitHub.

Can I specify the status when linking a work item to a GitHub commit or PR?

No. This feature isn't supported at this time.

Configuration and customizations

What is configurable or customizable?

Configuration and customization of Azure Boards occurs at the project and team level. For an overview of what you can configure and customize to support your business needs, see Configuration and customization of Azure Boards.

For FAQs on configuration and customization, see Azure Boards Configuration and Customization FAQs.

Related articles

- Query FAQs
- Work tracking, process, and project limits
- Azure Boards Configuration and Customization FAQs
- FAQs: Work in Excel connected to Azure Boards
- Work across projects FAQs
- Azure Boards extensions
- About teams and Agile tools

Feedback

Was this page helpful? Yes No

Provide product feedback

Naming restrictions and conventions

Article • 05/30/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Most components in Azure DevOps must follow naming restrictions and conventions. Restrictions help guarantee a consistent user experience and provide compatibility with other applications.

Common restrictions include not exceeding the character length for a name, not containing special characters, and maintaining uniqueness of names within an object set.

> **Important**
>
> When using the API, rather than user interface (UI), the API allows you to directly specify a name, which includes characters that might be restricted in the UI.
>
> **Best practices:**
>
> - Follow the UI restrictions to maintain consistency and prevent unintended issues.
> - Validate names programmatically and handle any special characters appropriately.

General considerations

The length restrictions in this article get measured by the number of Unicode characters permitted. Surrogate characters are composed of two Unicode characters, which count as two characters against the length restriction. For more information, see About Unicode and Character Sets.

As with other operating system files, ASCII control characters (ASCII 1-31) and surrogate combinations are also not allowed. For general information about the operating system restrictions applied to file names, see Naming Files, Paths, and Namespaces.

> **Note**
>
> - For limits on the number of items you can define, see **Work tracking, process, and project limits**.

System reserved names

Avoid using system reserved names, like the following examples:

- AUX
- COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, COM10
- CON
- DefaultCollection
- LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9
- NUL
- PRN
- SERVER, SignalR
- Web or WEB

For more information about reserved names, see File names, paths, and namespaces.

Azure Artifacts

Universal packages must conform to the following restrictions.

Expand table

Restriction type	Restriction
Package name	- Must be lowercase. - Must start and end with letters or numbers. - Must only contain letters, numbers, and nonconsecutive dashes, underscore, or periods.
Package version	Must be lowercase without build metadata.

Azure Boards

You use work items to capture information to plan and track your software development projects. With work items, you can describe the work to be done, assign work, track status, and coordinate efforts within your team. Different types of work items—such as user stories, tasks, bugs, and issues—track different types of information. For more information, see Azure Boards documentation.

All work item tracking objects are associated with one or more names. Most have friendly display names, except work item types and global lists, which are associated with reference names. A friendly name is a unique, user-visible identifier for a field. Using friendly names ensures consistency across projects and work item types in a project collection. The system uses the reference name internally and you can't change it once defined.

There are restrictions on several elements that are associated with work items, including reference and friendly names, field names, and attachment size.

Area and iteration paths

Two work item fields, **Area Path** and **Iteration Path**, provide a tree structure hierarchy for grouping work. Area paths group work items by product, functional, or feature area. Iteration paths group work items into sprints, milestones, or time periods for addressing those work items.

These multi-node fields use the backslash `\` characters to denote the hierarchy of nodes within the tree structure.

The names you assign to child nodes to these fields must conform to the following restrictions.

Expand table

Restriction type	Restriction
Node length	Must not contain more than 255 characters.
Reserved names	- Must not consist only of a period `.` or two periods `..`. - Must not be a system-reserved name, such as PRN, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, COM10, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9, NUL, CON, or AUX. For more information about reserved names, see File Names, Paths, and Namespaces.
Special characters for nodes	- Must not contain Unicode control characters. - Must not contain any one of the following characters: `\ / : * ? " < > | # $ * +`. - Must not contain characters prohibited by the local file system. For more information about Windows character restrictions, see Naming Files, Paths, and Namespaces.
Path length	Must not contain more than 4,000 Unicode characters.
Path hierarchy depth	Must be fewer than 14 levels deep.

Attachments

Files attached to work items must conform to the following restrictions.

Expand table

Restriction type	Restriction
File size	Must not exceed the maximum size: - Default maximum size: 4,096 kilobytes. - Absolute maximum size: 2 gigabytes.

Board column and swimlane names

Your board turns your backlog into an interactive signboard, providing a visual flow of work. As work progresses from idea to completion, you update the items on the board. Each column represents a work stage, and each card represents a user story (blue cards) or a bug (red cards) at that stage of work.

Customize your board by adding, removing, or renaming columns and swimlanes. Columns support the flow of work across the board. Swimlanes allow you to manage different classes of work as horizontal lanes on the board.

Column and swimlane names must conform to the following restrictions.

Expand table

Restriction type	Restriction
Length	Must not contain more than 256 Unicode characters.
Uniqueness	- Column names must not be the same as any other column name on the board. - Swimlane names must not be the same as any other swimlane name on the board.
Special characters	Must not contain any Unicode control characters or surrogate characters.

Field names

Each work item type contains one or more work item fields. These fields define the information stored for work items based on the work item type. Each work item field has an associated field reference name. The field reference name uniquely identifies each field and can't be changed once assigned. For more information about out-of-box work item fields, see Work item field index.

Work item field names must conform to the following restrictions.

Expand table

Restriction type	Restriction
Length	Must not contain more than 128 Unicode characters.
Uniqueness	- Field names must contain at least one alphabetic character. - Must not contain any one of the following characters: `. , ; ' : ~ \ / \ * ? " & % $ ! + = () [] {} <> -` |. - Must not contain leading or trailing spaces. - Must not contain two or more consecutive spaces.
Special characters	Must be unique within the organization or project collection. Work item field names are scoped to the project collection. If you rename a field name, you change it for all work items and WITs defined within all projects in the collection.

Field reference names and portability

The work item type definition language includes the concept of a *field reference name*. Field reference names can help you to port definitions between Team Foundation project collections and also to allow non-Microsoft integrations to find and refer to specific fields. These names are globally unique, just as a namespace in the .NET Framework application is globally unique.

The **System** namespace is used only to define all core system fields that are mandatory for Team Foundation system functions. You can't create your own System.X field because it might impede functionality.

The **Microsoft** namespace is used to define work item tracking fields. These fields are defined in a work item type definition of the process templates.

Customers and partners can create their own field namespaces for custom work item types. For descriptions of system fields and fields defined in the default process templates, see Index of work item fields.

ⓘ Important

- Field reference names can't be renamed. If for example, you changed the field name "Title" to "Header", the field reference name of that field remains the same. Integrations and internal representations of fields should use the field reference name instead of depending on the field name itself.
- Although you can create your own Microsoft.X field, we don't recommend that you do so, as it might impede functionality or the ability for the Configure Features wizard to successfully update a project after an upgrade.

Examples of field reference names

The following examples show valid field reference names, in various namespaces. Customers and partners can also define their own namespaces to support their custom work item types.

Expand table

System namespace examples	Microsoft namespace examples	Other namespace examples
System.Id, System.Title, System.CreatedBy, System.CreatedDate, System.ChangedBy, System.ChangedDate, System.State, System.Reason	Microsoft.VSTS.Build.FoundIn, Microsoft.VSTS.Common.Activity, Microsoft.VSTS.Common.Discipline, Microsoft.VSTS.Common.Priority, Microsoft.VSTS.CMMI.TaskType, Microsoft.VSTS.TCM.AutomationStatus, Microsoft.VSTS.TCM.TestSuiteType	The fictitious company, Fabrikam Fiber, might define the following custom work item fields: FabrikamFiber.Common.Severity, FabrikamFiber.Common.Phase, FabrikamFiber.RiskManagement.RiskType, FabrikamFiber.RiskManagement.Resolution The fictitious software company Contoso Corporation might define the following work item fields: Contoso.Common.BusinessPriority, Contoso.Bug.FoundInPhase, Contoso.Bug.FixInPhase

Work item tags

Work item tags correspond to one or two keyword phrases to filter work tracking tools such as backlogs, boards, and queries or define queries. For more information, see Add work item tags to categorize and filter lists and boards.

Tag names must conform to the following restrictions.

Expand table

Restriction type	Restriction
Length	- Must not contain more than 400 Unicode characters. - Must not be null or empty.
Special characters	- Must not contain separator characters: `,` or `;`. - Must not contain Unicode control or format characters, such as a line feed, paragraph separator, carriage return, or other mismatched surrogate characters.
Scope	Since global lists are available among all projects, they must not contain elements defined at the project level, such as project-specific group account definitions.

Azure Pipelines

Azure Pipeline definitions must conform to the following restrictions.

Expand table

Restriction type	Restriction
Job name	- Must only contain alphanumeric characters and `'_'`. - Must not start with a number. - Must have a unique name. - Must not contain keywords, for example: "deployment."
Stage name	- Must only contain alphanumeric characters and `'_'`. - Must not start with a number. - Must not contain keywords, for example: "deployment."
Expressions	Must start with `a-Z` or `_AND`, followed by `a-Z`, `0-9`, or `_`.

For more information, see Azure Pipelines documentation.

Azure Repos (Git)

Each Azure DevOps project can contain multiple Git repositories. The names you assign to Git repositories must conform to the following restrictions. To learn more, see Azure Repos Git documentation. For more information on naming restrictions for other Git items such as branches and tags, see git check-ref-format .

> **Important**
>
> Although you can include spaces within repo names, we don't recommend that you do so.

Expand table

Restriction type	Restriction
Length	Must not contain more than 64 Unicode characters.
Uniqueness	Must not be identical to any other Git repo name in the project.
Special characters	- Must not contain any Unicode control characters or surrogate characters. - Must not contain the following printable characters: `\ / : * ? " < > ; # $ * { } , + = [ ]` |. - Must not start with an underscore `_`.

Restriction type	Restriction
	- Must not start or end with a period `.`. - Must not be a system reserved name.

Azure Repos (TFVC)

Team Foundation version control (TFVC) provides a central repository for files and the commands that are required to manage those files across a team. It also provides customizable check-in policies, branching, merging, shelving, and many other features. To learn more, see Azure Repos TFVC documentation

Version control paths must conform to the following restrictions. See also Optimize your workspace.

Expand table

Restriction type	Restriction
Server source control folder path length	- Must not contain more than 259 Unicode characters for a single folder or file name. - Must not contain more than 388 Unicode characters for a directory. - Must not contain more than 399 Unicode characters for a combined directory and file name.
Local folder path length	- Must not contain more than 248 Unicode characters for a directory. - Must not contain more than 260 Unicode characters for a combined directory and file name.
Web portal and REST-specific length restrictions	Must not contain more than 215 Unicode characters for a combined project name, directory, and file name.

In the web portal and REST API, certain operations have a more restrictive path length than Visual Studio and TF.exe clients. These operations include Edit, Rename, and Delete, and occur because the web portal/REST uses a randomly generated workspace for concurrency when pending the changes.

TFVC files

The version control system stores many different types of files. Set up Team Foundation version control on your dev machine provides details on how to add existing Visual Studio projects or solutions.

Files and folders you add to Team Foundation version control must conform to the following restrictions.

Expand table

Restriction type	Restriction
Files and folders	- Names must not contain the following printable characters: `\ / : * ? " < > ;`. - Folders must not be outside the mapped directory for the active workspace.
File names	- Must not begin with a `$`. - Must not contain the following printable characters: `\ / : * ? " < > ;` |.

TFVC labels

In Team Foundation version control (TFVC), a label is a name applied to a specific set of revisions. You can attach labels to a set of unrelated files in version control, which lets you retrieve the files or act upon them as a group. The following table describes the restrictions put on label names.

Expand table

Restriction type	Restriction
Length	Must not contain more than 64 Unicode characters.
Special characters	- Can't end with a space `( )` or period `.`. - Must not contain the following printable characters: `\ / : * ? " < > ; @` |.

TFVC Shelvesets

Shelvesets enable you to set aside temporarily a batch of pending changes and then, as an option, remove the pending changes from your workspace. Later, you can restore the changes in a shelveset to your workspace or put them into another user's workspace.

Shelveset names must conform to the following restrictions.

Expand table

Restriction type	Restriction
Length	Must not contain more than 64 Unicode characters.
Special characters	Must not contain the following printable characters: `\ / : * ? " < > & ;` |.

TFVC workspaces

A workspace is a client-side copy of the files and folders in Team Foundation version control (TFVC). When you create multiple workspaces, you can have different versions of the same version control folder on a client computer. Create and work with workspaces provides more details.

Workspace names must conform to the following restrictions.

Expand table

Restriction type	Restriction
Length	Must not contain more than 64 Unicode characters.
Special characters	- Must not end with a space `( )`. - Must not contain the following printable characters: `\ / : * ? " < > ;` |.

Organization names

> **Important**
>
> Adhere to the following guidelines when you create an organization name.
>
> - Use only letters from the English alphabet
> - Start your organization name with a letter or number
> - Use letters, numbers, or hyphens after the initial character
> - Ensure that your organization doesn't exceed 50 Unicode characters
> - End with a letter or number
>
> If you use any of the disallowed characters, you get the following error message: VS850015: The specified name is not allowed to be used: {Organization name}.

Process and process templates

A process defines the building blocks of the work item tracking system and other subsystems that you have access to after connecting to a project.

> **Note**
>
> **Terminology**: Both "process" and "process template" refer to an interdependent set of files used to create a project. For more information, see About processes and process

templates, which describes the differences among the three default processes available to you.

Processes that you define or customize must conform to the following restrictions.

Expand table

Restriction type	Restriction
Length	Must not contain more than 256 Unicode characters.
Uniqueness	- Must be unique across Azure DevOps. - If you upload a template with the same name as an existing template, the existing template is overwritten.
Process template file size	Must not exceed 2 GB (gigabytes).

Project names

A project establishes a repository for source code and a place for teams to plan, track progress, and collaborate. The name of the project is part of the connection string used to connect team members to projects.

Names you assign to projects that you create must conform to the following restrictions.

Expand table

Restriction type	Restriction
Length	Must not contain more than 64 Unicode characters.
Uniqueness	Must not be identical to any other name in the project collection, the SharePoint Web application that supports the collection, or the instance of SQL Server Reporting Services that supports the collection.
Reserves names	- Must not be a system reserved name. - Must not be one of the hidden segments used for IIS request filtering like App_Browsers, App_code, App_Data, App_GlobalResources, App_LocalResources, App_Themes, App_WebResources, bin, or web.config.
Special characters	- Must not contain any Unicode control characters or surrogate characters. - Must not contain the following printable characters: `\ / : * ? " < > ; # $ * { } , + = [ ]` |. - Must not start with an underscore `_`. - Must not start or end with a period `.`.

Security groups

With Azure DevOps security groups, you can apply certain rights or permissions to a group of users.

On-premises groups might consist of Active Directory group accounts, Azure DevOps security groups, Windows user accounts, Windows group accounts, or any mixture of these types. For more information, see Add Active Directory / Microsoft Entra users or groups to a built-in security group.

Security groups must conform to the following restrictions.

Expand table

Restriction type	Restriction
Security group account name length	Must not contain more than 256 Unicode characters.
Uniqueness	- Project-level group accounts must not match any other group name in the same project. - Collection-level group accounts must not match any other name of a group account in the project collection.
Reserved group names	Must not be named with a $NAMESPACE at either the project or the server level.
Special characters	- Must not contain any Unicode control characters or surrogate characters. - Must not contain the following printable characters: `/ \ [ ] : < > + = ; ? *` |. - Must not include nonprintable characters in the ASCII value range of 1-31. - Must not end in a period `.`. - Must not include commas `,`.

Team names

Team names identify a group of individuals or groups that collectively work together as a team in a project. Team members use this name to connect to the team or to query against members defined for a team.

Team names must conform to conventions that allow them to be rendered as part of a valid URL. Each team name must be unique within a single project. However, there aren't any restrictions on using the same team name in different projects within a project collection. For more information, see Add another team or a hierarchy of teams.

Team names must conform to the following restrictions.

Expand table

Restriction type	Restriction
Length	Must not contain more than 64 Unicode characters.
Uniqueness	Must not be identical to any other name in the project.
Reserved names	Must not be a system reserved name.
Special characters	- Must not contain any Unicode control characters or surrogate characters. - Must not contain the following printable characters: `\ / : * ? " < > ; # $ * { } , + = [ ]` |. - Must not contain an ellipsis `...` or a double period `..`. - Must not start with an underscore `_`. - Must not start or end with a period `.`.

User account names

User accounts identify people added to a project or project collection. User accounts might correspond to entries within Active Directory, Microsoft Entra ID, Windows server group.

User accounts that you add to an organization or collection must conform to the following restrictions. To add user accounts to a project, see Add users to a project or team.

Expand table

Restriction type	Restriction
Account name length	Must not contain more than 256 Unicode characters.
Uniqueness	Must not match any other user account added to the project collection.
Reserved group names	Must not be named with a `$NAMESPACE` at either the project or the server level.
Special characters	- Must not include the following printable characters: `"/ \ [ ] : < > + = ; ? *` |. - Must not include nonprintable characters in the ASCII value range of 1-31. - Must not end in a period `.` or a dollar sign `$`. - Must not include commas `,`. - Must not include the following Unicode categories: LineSeparator, ParagraphSeparator, Control, Format, OtherNotAssigned.

Wiki page and file names

Each wiki page corresponds to a file within the wiki Git repo. Names you assign to a wiki page title must conform to the following restrictions.

Expand table

Restriction type	Restriction
File name	The fully qualified page path shouldn't exceed 235 characters.
Uniqueness	Page titles are case sensitive and must be unique within the wiki hierarchy.
Special characters	- Must not contain any Unicode control characters or surrogate characters. - Must not contain the following printable characters: `/ \ #`. - Must not start or end with a period `.`.
File size	Must not exceed the maximum of 18 MB.
Attachment file size	Must not exceed the maximum of 19 MB.

Special characters in Wiki page titles

You can specify page titles which contain one or more of these special characters `: < > * ? | -`. For example, you can name a Markdown file as "FAQ?" or "Set-up guide". The characters have the following URI encoded strings:

Expand table

Character	Encoded string
`:`	%3A
`<`	%3C
`>`	%3E
`*`	%2A
`?`	%3F
`|`	%7C
`-`	%2D
`"`	%22

Related articles

- Azure Artifacts count and size limits
- Work tracking, process, and project limits
- Customize work tracking objects to support your team's processes
- Customize a process template

Feedback

Was this page helpful? Yes No

Provide product feedback

Work tracking, process, and project limits

Article • 09/06/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

This article defines operational and object limits placed on work tracking operations and work tracking customization. In addition to the specified hard limits on specific objects, some practical limits apply. When you customize work item types (WITs), consider the limits placed on objects.

Work items and queries

When you define work items or running queries, keep the following operational limits in mind:

Expand table

Object	Limit
Attachments added to a work item	100
Attachment size	60 MB
Long text field	1-M characters
Query execution time	30 seconds
Query results	20,000 items
Query length	32,000 characters
Shared queries under a folder	999 queries
Work item links assigned to a work item	1,000
Work item tags assigned to a work item	100
Work item revisions (REST API)	10,000
Favorite queries per project	200 queries

The REST API for Azure DevOps Services enforces a work item revision limit of 10,000 updates. This limit restricts updates made through the REST API, but updates from the web portal aren't affected.

To improve query performance, see Define a query/Best practices.

Backlogs, boards, dashboards, and teams

When you work with teams, work item tags, backlogs, and boards, the following operational display and object limits apply.

Expand table

User interface	Limit
Backlogs	10,000 work items
Boards	1,000 cards (excluding those cards in the *Proposed* and *Completed* workflow state categories)
Taskboard	1,000 tasks
Area Paths	10,000 per project
Area Path Depth	14
Area Paths per team	300
Iteration Paths	10,000 per project
Iteration Path Depth	14
Iteration Paths per team	300
Project Dashboards	500 per project. Accessible at the project level and anyone with access to the project can use.
Team Dashboards	500 per team. Specific to the team and used to track team-specific metrics and data.
Teams	5,000 per project
Work item tags	150,000 tag definitions per organization or collection
Delivery plans per project	1,000
Templates per work item type	100

Each backlog can display up to 10,000 work items. This limit applies to what the backlog can display, not to the number of work items you can define, as there's no specific limit.

If your backlog exceeds this limit, consider adding a team and moving some work items to the new team's backlog.

> **Tip**
>
> If you're approaching the dashboards limits, see the following steps to manage and clean up your dashboards:
>
> - **Review usage:** Identify dashboards that are no longer in use or are duplicates. You can do this by checking the last accessed date or by consulting with team members.
> - **Consolidate dashboards:** Combine similar dashboards to reduce the total number. This can be done by adding multiple widgets to a single dashboard.
> - **Archive old dashboards:** If certain dashboards are no longer needed but you want to keep the data, consider exporting the data and archiving the dashboards.
> - **Use the Object Limit Tracker feature:** Provides real-time visibility into resource usage, including dashboards. This feature can help you **proactively manage your limits and avoid potential issues** .

Other notes:

- Completed or closed work items don't display on backlogs and boards once their **Changed Date** is older than a year. You can still list these items using a query. To make them show up on a backlog or board, make a minor change to reset the display clock.
- Avoid nesting backlog items of the same type. For more information, see Fix reordering and nesting issues.
- Avoid assigning the same area paths to more than one team. For more information, see Limitations of multi-team board views.
- By default, work item limits might be set to lower values initially.

Projects

Azure DevOps Services limits each organization to 1,000 projects per organization, an increase over the previous limit of 300 projects.

> **Note**

Above 300 projects, certain experiences, like connecting to a project from Visual Studio, might degrade. For on-premises Azure DevOps Server, there are no hard limits, but performance issues may arise as the number of projects nears 300. When migrating to Azure DevOps Services, observe the maximum limit of 1,000 projects. If your collection exceeds this limit, split the collection or delete older projects.

For more information, see Migrate data from Azure DevOps Server to Azure DevOps Services.

Process customization

Many limits are imposed on the number of objects you can define for a process. For more information, see Customize your work tracking experience.

The following table lists the maximum number of objects you can define for the Inheritance and Hosted XML process models. While these limits are hard limits, practical limits might also apply.

Expand table

Object	Inheritance	Hosted XML
Number of processes you can have in an organization	128	64
Work item types defined for a process	64	64
Fields defined for an organization	8192	8192
Fields defined for a process	1024	1024
Fields defined for a work item type	1024	1024
Picklists defined for an organization or collection	2048	-
Picklist items defined for a list	2048	2048
Picklist item character length	256	-
Workflow states defined for a work item type	32	16
Rules defined for a work item type	1024	1024
Actions defined for a rule	10	10
Portfolio backlog levels defined for a process	5	5
Categories defined for a process	-	32

Object	Inheritance	Hosted XML
Global lists defined for a process	-	256
List items defined within a global list	-	1024
Work item attachment size	60 MB	60 MB

For other restrictions and conformance requirements of the Hosted XML process model, see Customize a process when using Hosted XML.

> ⓘ Note
>
> For the Hosted XML process model, you can define approximately 10,000 items across all global lists specified in all WITs.

Practical limits

To minimize performance issues, we recommend following this guidance:

- Limit the number of custom fields you define. All custom fields contribute to the total allowed for a process, collection, or organization. You can specify different behaviors, such as rules and picklists, for the same field in different WITs.
- Limit the number of rules you define for a WIT. While you can create multiple rules for a WIT, other rules can negatively affect performance when users add or modify work items. When users save work items, the system validates all rules associated with the fields for that work item type. In some cases, the rule validation expression might be too complex for SQL to evaluate efficiently.
- Limit the number of custom WITs you define.

> ⓘ Note
>
> **Work Item Rules Validation Exceeds SQL Limits**: A single SQL expression is defined per project to validate work items whenever they are created or updated. This expression grows with the number of rules specified for all work item types in the project. Each behavioral qualifier for a field increases the number of sub-expressions. Nested rules, rules that apply only on a transition, or rules conditioned on the value of another field add more conditions to an IF statement. Once the expression reaches a certain size or complexity, SQL can no longer evaluate it and generates an error. To resolve this error, remove some WITs or eliminate some rules.

Rate limits

To reduce costs and enhance scalability and performance, Azure DevOps Services, like many Software-as-a-Service solutions, uses multi-tenancy. To ensure good performance and minimize the risk of outages, Azure DevOps Services limits the resources individuals can consume and the number of requests they can make to certain commands. When these limits are exceeded, subsequent requests might be delayed or blocked.

Most rate limits are reached through REST API calls or nonoptimized queries. For more information, see Rate limits and Best practices (to avoid hitting rate limits).

Migrate and import limits

When migrating from on-premises to Azure DevOps Services, you might encounter several size limits, including:

- Database size exceeding the recommended size
- Largest table size exceeding the recommended size
- Database metadata size exceeding the supported size

For more information, see Migrate data from Azure DevOps Server to Azure DevOps Services and Troubleshoot import and migration errors.

Related articles

- Learn about process customization and inherited processes
- Create an inheritance process
- Follow best practices
- Adhere to naming restrictions and conventions

Related resources

- Tags Manager
- WIQL Editor

Feedback

Was this page helpful? Yes No

Provide product feedback

Field descriptions for default and work item fields used in process templates

Article • 08/20/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Use this index to find descriptions of each field used to track work items. This reference covers all fields defined within the core system processes/templates: Basic, Agile, Scrum, and CMMI. The fields and work item types (WITs) available to you depend on the process you selected when you created your project.

To support other tracking needs, define your own custom work item fields.

> ⓘ **Note**
>
> The Analytics Service doesn't support reporting on plain text and HTML fields.

Alphabetical index

Values in parenthesis indicate the following criteria:

- **System**: Core system field assigned to all work item types for all processes
- **Agile**: Used only by the Agile process
- **CMMI**: Used only by the CMMI process
- **Scrum**: Used only by the Scrum process
- **TCM**: Used to support test case management

A

- Acceptance Criteria (Scrum)
- Accepted By
- Accepted Date
- Activated By
- Activated Date
- Activity
- Actual Attendee 1-8 (CMMI)
- Analysis (CMMI)
- Application Launch Instructions

- Application Start Information
- Application Type
- Area ID (System)
- Area Path (System)
- Assigned To
- Associated Context
- Associated Context Code
- Associated Context Owner
- Associated Context Type
- Attached File Count
- Authorized As (Not used)
- Automated Test ID (TCM)
- Automated Test Name (TCM)
- Automated Test Storage (TCM)
- Automated Test Type (TCM)
- AutomatedTestId (TCM)
- AutomatedTestName (TCM)
- Automation Status (TCM)

B

- Backlog Priority (Scrum)
- Blocked
- Board Column
- Board Column Done
- Board Lane
- Business Value

C

- Called By (CMMI)
- Called Date (CMMI)
- Changed By (System)
- Changed Date (System)
- Closed By (System)
- Closed Date (System)
- Closed Status
- Closed Status Code
- Closing Comment
- Comment Count

- Comments (CMMI)
- Committed (CMMI)
- Completed Work
- Contingency Plan (CMMI)
- Corrective Action Actual Resolution (CMMI)
- Corrective Action Plan (CMMI)
- Created By (System)
- Created Date (System)

D-E-F

- Discipline (CMMI)
- Description (System)
- Due Date
- Effort
- Escalate (CMMI)
- External Link Count
- Finish Date
- Found In Build (TCM)
- Found In Environment (CMMI)

H

- History (System)
- How Found (CMMI)
- Hyperlink Count

I

- ID (System)
- Impact Assessment (CMMI)
- Impact on Architecture (CMMI)
- Impact on Development (CMMI)
- Impact on Technical Publications (CMMI)
- Impact on Test (CMMI)
- Impact on User Experience (CMMI)
- Integrated in Build (TCM)
- Issue (TCM)
- Iteration ID (System)
- Iteration Path (System)

J-L-M-N

- Justification (CMMI)
- Link Comment (System)
- Link Description (System)
- Local Data Source (TCM)
- Meeting Type (CMMI)
- Minutes (CMMI)
- Mitigation Plan (CMMI)
- Mitigation Triggers (CMMI)
- Node Name (System)

O-P-Q

- Optional Attendee 1-8 (CMMI)
- Original Estimate
- Parameters (TCM)
- Parent[1]
- Priority
- Probability (CMMI)
- Proposed Fix (CMMI)
- Purpose (CMMI)
- Query Text (TCM)

R

- Rating
- Reason (System)
- Related Link Count (System)
- Remaining Work
- Remote Link Count[2] (System)
- Repro Steps
- Required Attendee 1-8 (CMMI)
- Requirement Type (CMMI)
- Requires Review (CMMI)
- Requires Test (CMMI)
- Resolution (Scrum)
- Resolved By
- Resolved Date
- Resolved Reason

- Rev (System)
- Reviewed By
- Reviewed Date
- Revised Date (System, TCM)
- Risk (Agile)
- Root Cause (CMMI)

S

- Severity
- Size (CMMI)
- Stack Rank
- Start Date
- State (System)
- State Change Date
- State Code
- Steps (TCM)
- Steps to Reproduce (TCM)
- Story Points (Agile)
- Subject Matter Expert (CMMI)
- Symptom (CMMI)
- System Info (TCM)

T

- Tags
- Target Date
- Target Resolve Date (CMMI)
- Task Type (CMMI)
- Team Project (System)
- Test Suite Audit (TCM)
- Test Suite Type (TCM)
- Test Suite Type ID (TCM)
- Time Criticality
- Title (System)
- Triage (CMMI)

U-V-W

- User Acceptance Test (CMMI)

- Value Area
- Watermark (System)
- Work Item Type (System)

> **Note**
>
> 1. This field is available from Azure DevOps Services and Azure DevOps Server 2020 and later versions.
> 2. This field is available for Azure DevOps Services only.

Use system fields or custom fields added to your project collection to enable meaningful cross-project reports and queries. Any nonsystem field referenced in the workflow or forms section of the work item type definition must have a **FIELD** element defined in the **FIELDS** section of the work item type definition XML file. Specify any nonsystem field you want to use for generating queries or reports in the **FIELDS** section.

Field reference articles

The following articles describe fields that are commonly used by several WITs or are functionally specific to one or a few WITs. For more information, see Work item field and attributes in Azure Boards.

Fields common to many work types

- Titles, IDs, and descriptive fields
- History and revision changes
- Areas and iterations
- Assignments and account-specific fields
- Planning, ranking, and priorities
- Work estimates, activity, and other numeric fields
- Build and test integration fields
- Links and attachment related fields

Fields used by specific work item types

- Code Review Request
- Code Review Response
- Feedback Request
- Feedback Response

- Shared Steps
- Test Case

Fields used to track CMMI work items

- Requirements
- Bugs
- Change Requests
- Issues
- Review Meetings
- Risks

Related articles

- Learn about work item fields
- Learn about managed queries
- Define a query
- Learn about processes and process templates

Feedback

Was this page helpful? Yes No

Provide product feedback

List work item fields and attributes

Article • 11/22/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Work item fields are used to track information within Azure DevOps. These fields are defined at the organization level and are shared across all projects in the organization. To review the fields defined for your organization, you can use one of the following two tools, available for both Inherited and Hosted XML process models:

- Process > Fields web page
- Work Item Field Explorer

For a description of each field defined with a system process, see Work item field index.

Prerequisites

Permissions: To view the fields defined for an organization or collection, be a member of the **Project Collection Valid Users** application group or have the **View instance-level information** permission set to **Allow** for the organization or collection.

List or review fields

To list or review fields, use one of the following tools based on your process model—**Inherited**, **Hosted XML**, or **On-premises XML**. For an index of fields defined within the default processes, see Work item field index.

Expand table

Tool	Inheritance	Hosted XML	On-premises XML
Web portal: List inherited and custom-defined fields	✓	✓1	
Work item field explorer	✓	✓	✓
witadmin listfields command line tool	✓	✓	✓

> **Note**

1. Only supported for default processes (Agile, CMMI, Scrum).

Field data types and names

Each work item type specifies the fields included in work items of that type. Each field has multiple attributes, many of which are system-defined and immutable.

Fields get defined by the following three attributes:

- **Data type**: Indicates the kind of data that can be entered into the field, such as Boolean, Double, Integer, HTML, or String. For detailed descriptions of each data type, see Query fields, operators, and macros.
- **Friendly name**: The user-friendly name assigned to the field, used when selecting a **Field** in a query clause. This name might differ from the name displayed on the work item form.
- **Reference name**: The identifier used when creating WIQL queries, improvised work item templates, executing REST API commands, or defining XML work item type definitions. The reference name is permanent and can't be changed once set.

For a detailed description of each field attribute and instructions on how to list them, see Field attributes and List field attributes later in this article. For an overview of work item types (WITs) and work items, refer to Track work with user stories, issues, bugs, features, and epics.

What is a field? How are field names used?

Each work item type includes 31 system fields and several type-specific fields. Work items are used to plan and track your project.

Each field captures specific information about the work to be performed. The values you assign to these fields are stored in the work tracking data store, enabling you to create queries that determine status and trends.

For descriptions and usage of each field defined for the core system processes—Agile, Basic, Scrum, and CMMI—refer to the Work item field index.

Field names

A work item field name uniquely identifies each work item field. Ensure your field names adhere to the following guidelines:

- **Uniqueness**: Field names must be unique within the account or project collection.
- **Length**: Field names must contain 128 Unicode characters or fewer.
- **Spacing**: Field names can't have leading or trailing spaces or contain two or more consecutive spaces.
- **Composition**: Field names must include at least one alphabetic character.
- **Prohibited Characters**: Field names can't include the following characters: `.,;'`:~/*|?"&%$!+=()[]{}<>`.

Since custom fields are defined at the organization or collection level, you can't add a custom field with the same name to multiple processes.

For more information, see Naming restrictions and conventions.

System and predefined fields

A work item field name uniquely identifies each field within Azure DevOps. Ensure your field names comply with the following guidelines:

- **Uniqueness**: Field names must be unique within an account or project collection.
- **Length**: Field names must be 128 Unicode characters or fewer.
- **Spacing**: Field names can't have leading or trailing spaces or contain multiple consecutive spaces.
- **Composition**: Field names must include at least one alphabetic character.
- **Prohibited Characters**: Field names can't include the following characters: `.,;'`:~/*|?"&%$!+=()[]{}<>`.

Since custom fields are defined at the organization or collection level, you can't add a custom field with the same name to multiple processes.

For more information, see Naming restrictions and conventions.

Custom fields

Because custom fields are defined at the organization or project collection level, you can't add a custom field with the same name to multiple processes.

When adding custom fields, note the following limits:

- **Maximum fields per work item type (WIT):** 64
- **Maximum fields per process:** 512

The field data type determines the kind and size of data that can be stored in the field. Each field can have only one type defined within a project collection, encouraging the

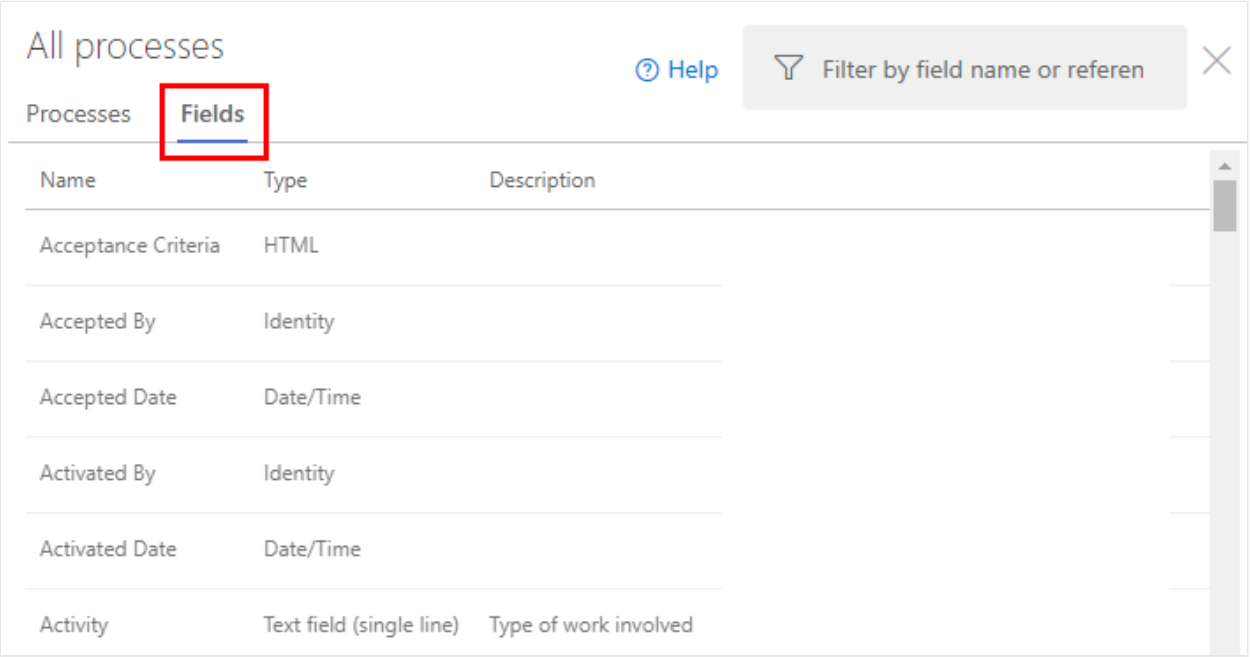
use of common fields across different projects and work item types.

When you add a custom field to an inherited process, Azure DevOps assigns a reference name prefixed with *Custom* followed by the field name without spaces. For example, adding a field named *DevOps Triage* results in the reference name **Custom.DevOpsTriage**. Spaces aren't allowed in reference names.

For more information, see Naming restrictions and conventions.

How can I determine the field data type?

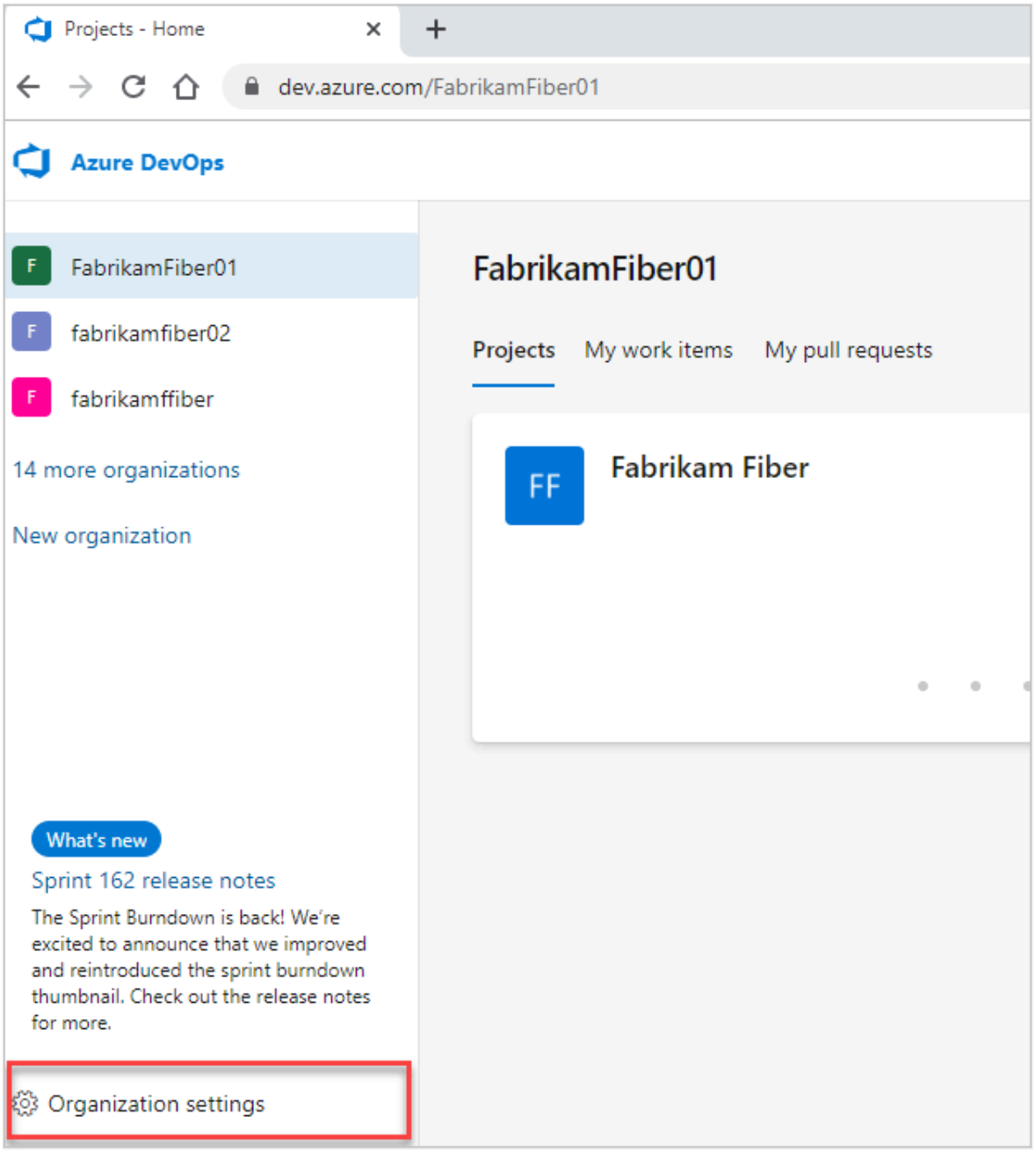
You can view the data type of fields defined for your organization by opening the Process>Fields page.



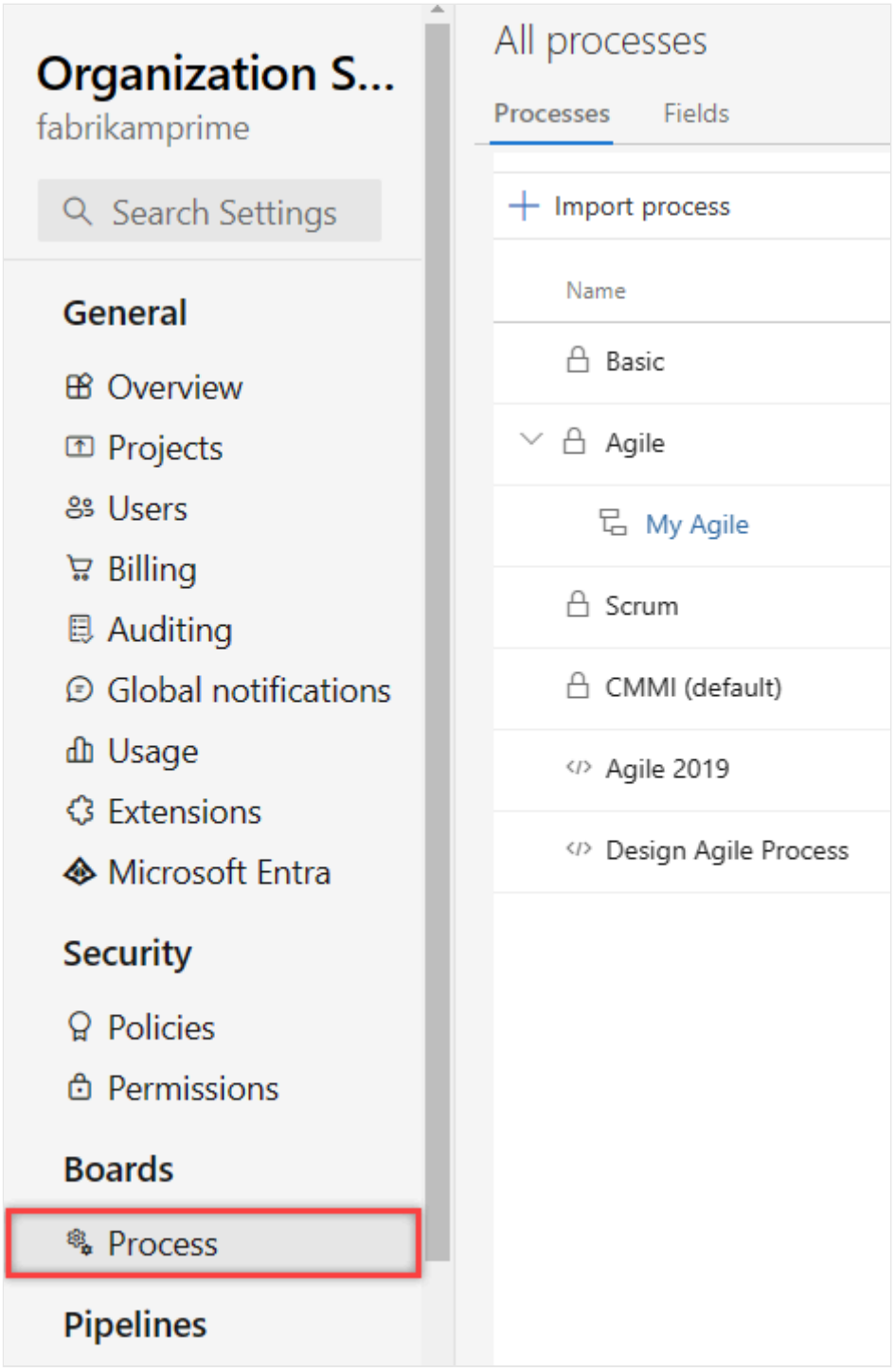
Review fields list

To review the list of fields defined for an organization or collection, do the following steps:

1. Select the Azure DevOps logo to open **Projects**. Then choose **Organization settings**.

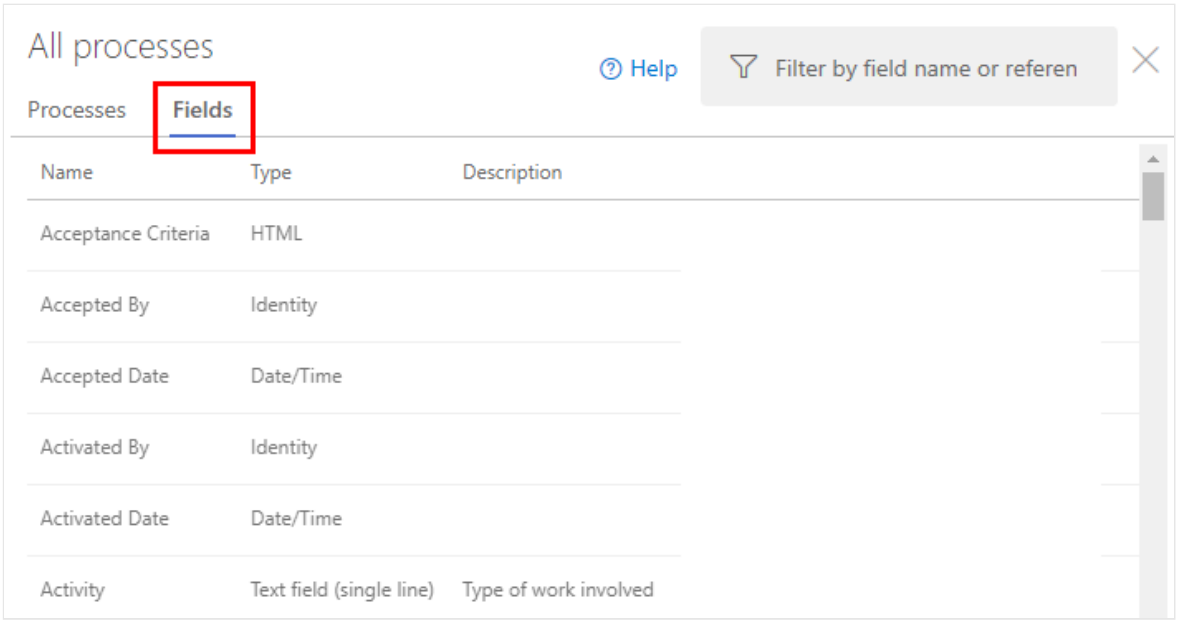


2. Select **Process**.



3. Select **Fields**.

   Fields listed correspond to all fields defined for the organization or collection, which includes all custom fields and the fields defined for system processes.

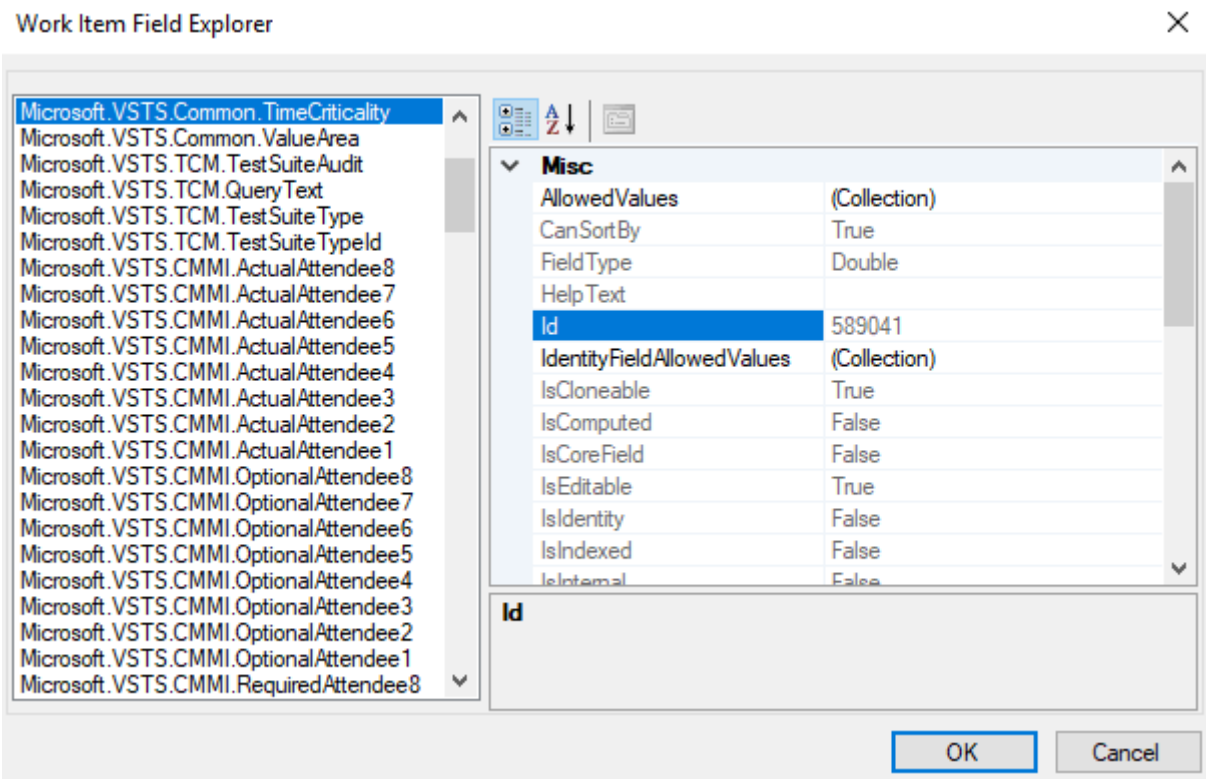


> **Note**
>
> If you don't see **Fields**, then your collection uses the On-premises XML process. The Fields page isn't supported for that process.

For descriptions, usage, and reference names of each field, refer to the Work item field index. Additionally, you can retrieve field reference names using the Work Item Types Field - List REST API.

Work Item Field Explorer

You can look up the assignments of field attributes using the Work Item Field Explorer tool.



To access the Work Item Field Explorer, install the Process Editor Tool. Based on your installed version of Visual Studio, get the Process Editor Tool from one of the following extensions.

- **Visual Studio 2019 & 2022:** As of now, there's no specific Process Template Editor extension available. But, you can still customize process templates using other tools such as any text editor or XML editor. You can also use the Azure DevOps web portal for certain customizations, especially for inherited processes.
- **Visual Studio 2017**: TFS Process Template Editor. You can also use this version of the Process Editor to modify the old-style work item forms. You can't use it to edit forms associated with the new web forms.
- **Visual Studio 2015**: TFS Power Tools.

Field attributes

There are many nonchangeable and hidden attributes for each work item field. The following table describes each attribute. Attributes have different names depending on whether you retrieve them through the **Fields - Get REST API** or view them through the Work Item Field Explorer (**WIFE**) tool.

Attributes assigned to a field depend on the platform and version you're using. For example, some attributes aren't supported with the Inheritance process. To look up the reference name for a field, see the Work item field index.

Attribute

Attribute type

Description

REST:
WIFE: **AllowedValues**

collection

Gets the collection of valid values for a field that contains picklist values. You can change this by specifying a picklist or global list (on-premises).
Can change?=Yes

REST: **canSortBy**
WIFE: **CanSortBy**

boolean

Indicates whether you can sort query results with this field.
Can change?=No

REST: **description**
WIFE: **HelpText**

string

Specifies a description for the field, which also defines the help text that appears when you hover over the field within the work item form.
Can change?=Yes

REST:
WIFE: **ID**

Integer

Specifies the internal ID of the field.
Can change?=No

REST:
WIFE: **IsCloneable**

boolean

Indicates whether the value defined for the field is copied when a user chooses to copy a work item. For example, **Title**, **Tags**, and **Description** fields are copied, but the **ID** and **History** fields aren't copied.
Can change?=No

REST:
WIFE: **IsComputed**

boolean

Indicates if the value set by this field is computed by the system (True) or not (False). Examples of computed fields are ones that are set by the system, such as the **ID**, **Revised Date**, **Changed Date**, and **External Link Count**.
Can change?=No

REST:
WIFE: **IsCoreField**

boolean

Indicates whether this field is specified for all work item types.
Can change?=No

REST:
WIFE: **IsEditable**

boolean

Indicates if users can modify this field (True) or not (False). Examples of noneditable fields are ones that are set by the system, such as the **ID**, **Revision**, **Created By**, and **Changed By** fields
Can change?=No

REST: **isIdentity**
WIFE: **IsIdentity**

boolean

Indicates whether this field is an *Identity* field. Identity fields are string fields used to store user identities.
Can change?=No

REST:
WIFE: **IsIndexed**[1]

boolean

Indicates whether this field is indexed to support search.
Can change?=No

REST:
WIFE: **IsLongText**

boolean

Indicates that the field can contain more than 255 characters, such as fields assigned a data type of *PlainText*, *HTML*, or *History*.
Can change?=No

REST: **isPicklist**[2] WIFE:

Boolean

Indicates whether the field is associated with a picklist. The value is set to *True* when a custom field is defined for Azure DevOps and Picklist (String) or Picklist (Integer) type is selected. The value is set to *False* for inherited fields that define picklists.
Can change?=No

REST: **isPicklistSuggested**[2] WIFE:

Boolean

Indicates whether the field allows users to enter their own values for a picklist. The value is set to *True* when a custom field is defined for Azure DevOps, Picklist (String), or Picklist (Integer) type is selected, and the checkbox for **Allow users to set their own values** is checked.
Can change?=Yes

REST: **isQueryable**
WIFE: **IsQueryable**

boolean

Indicates if the field shows up within the set of fields you can add to filter a work item query (True), or not (False). Most fields are queryable.

Can change?=No

REST:
WIFE: **IsReportable** [3]

boolean

Indicates if the reportable attribute is defined or set to anything other than **None**. This attribute can be changed for on-premises environments.
Can change?=Yes

REST:
WIFE: **IsUsedInGlobalWorkflow**

boolean

Indicates if the field is defined within a global workflow.
Can change?=No

REST:
WIFE: **IsUserNameField**

boolean

Indicates if the field is used to display an Identity field.
Can change?=No

REST: **name**
WIFE: **Name**

string

Friendly name assigned to the field. The friendly name can't be changed for Azure DevOps, but can be changed for on-premises using the **witadmin changefield** command.
Can change?=On-premises only

REST: **picklistId**
WIFE: **HelpText**

GUID

If the field is a picklist, the identifier of the associated picklist, otherwise null. A unique GUID value is assigned when a custom field is defined for Azure DevOps and Picklist

(String) or Picklist (Integer) type is selected.
Can change?=No

REST:
WIFE: **ProhibitedValues**

collection

Gets the collection of prohibited values for a field that specifies such values. You can only define prohibited values for on-premises deployments.
Can change?=On-premises only

REST: **readOnly**
WIFE:

Boolean

Indicates whether the field is set to read only. For Azure DevOps Services, only custom fields can be changed to be read-only. System fields can't be modified.
Can change?=Yes

REST: **referenceName**
WIFE: **ReferenceName**

string

Specifies the reference name of a field.
Can change?=No

REST:
WIFE: **ReportingAttributes**[3]

Specifies **Detail**, **Dimension**, or **Measure**, depending on whether and how you want the field to be included in reports. Data from fields that have a value other than **None** for this attribute are exported to the data warehouse and can be included in SQL reports.
Can change?=On-premises only

REST:
WIFE: **ReportingName**[3]

string

Specifies the label for a field when data appears in SQL reports. If you don't specify a value, the field's friendly name is used.
Can change?=On-premises only

REST:
WIFE: **ReportingReferenceName**[3]

string

Specifies a different reference name to a field that is used when data is exported to the relational data warehouse. If you don't specify a value, the fields reference name is used.
Can change?=On-premises only

REST: **supportedOperations**
WIFE:

set

The set of query operators that are valid for use when referencing this field. For a quick reference of supported operations based on data type, see Query quick reference, Operators, and macros supported for each data type.
Can change?=No

REST:
WIFE: **SupportsTextQuery**

boolean

Indicates whether the field supports text queries such as **Contains Words**, **Does Not Contains Words**.
Can change?=No

REST:
WIFE: **SystemType**

string

Specifies the data type of the field, referencing the system name such as System.DateTime or System.String.
Can change?=No

REST: **type**
WIFE: **FieldType**

string

Specifies the data type of the field, such as *Boolean*, *DateTime*, *Integer*, *String*, and so on. For a complete list and descriptions, see Query fields, operators, and macros.
Can change?=No

REST: **usage**
WIFE: **Usage**

string

Specifies whether the field is intended for use with work items (*WorkItem*) or work item link (*WorkItemLink*) objects. The usage for most fields is WorkItem. For a complete list of usage values, see Get Fields, FieldUsage.
Can change?=No

> **Note**
>
> 1. For on-premises deployments, you can enable indexing for a field to improve query response times when filtering on the field. For more information, see Indexed fields later in this article.
> 2. The **isPicklist** and **isPicklistSuggested** attributes are only assigned to custom fields defined for an inherited process. The Inherited process model is supported for Azure DevOps Server 2019 and later versions. For more information, see Inherited process model.
> 3. All reporting attributes are valid only for on-premises deployments whose projects have been configured to support SQL Server Reporting and SQL Server Analysis Services.

List field attributes

You can list the attributes assigned to a field by using the **Fields - Get REST API**. Replace *OrganizationName* with your actual organization name.

```
REST
```

```
https://dev.azure.com/OrganizationName/_apis/wit/fields/FieldReferenceName
```

For example, to list the attributes for the Iteration Path, use the reference name `System.IterationPath` for the `fabrikam` organization:

REST

```
https://dev.azure.com/fabrikam/_apis/wit/fields/System.IterationPath

**Returned data:**

> [!div class="tabbedCodeSnippets"]
> ```JSON
> {
> "name": "Iteration Path",
> "referenceName": "System.IterationPath",
> "description": "The iteration within which this bug will be fixed",
> "type": "treePath",
> "usage": "workItem",
> "readOnly": false,
> "canSortBy": true,
> "isQueryable": true,
> "supportedOperations": [
> {
> "referenceName": "SupportedOperations.Under",
> "name": "Under"
> },
> {
> "referenceName": "SupportedOperations.NotUnder",
> "name": "Not Under"
> },
> {
> "referenceName": "SupportedOperations.Equals",
> "name": "="
> },
> {
> "referenceName": "SupportedOperations.NotEquals",
> "name": "<>"
> },
> {
> "referenceName": "SupportedOperations.In",
> "name": "In"
> },
> {
> "name": "Not In"
> }
> ],
> "isIdentity": false,
> "isPicklist": false,
> "isPicklistSuggested": false,
> "url": "https://dev.azure.com/mseng/_apis/wit/fields/System.IterationPath"
> }
```

```
> ```
```

Add and modify fields

To add fields to a process, you add them to one or more work item types. For more information, see Customize an inheritance process.

Related articles

- Refer to the Query quick reference to quickly understand query syntax and usage.
- Explore the Work item field index to view all available work item fields.
- Add and manage fields for an inherited process to customize fields according to your inherited process requirements.
- Access the metadata reference for Azure Boards Analytics to understand analytics metadata and reporting capabilities.

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Feedback

Was this page helpful? Yes No

Provide product feedback

About workflow states in backlogs and boards

Article • 11/11/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

Workflows are essential for managing work items, consisting of states, transitions, and reasons. Each workflow is defined for a specific work item type. Transitions allow movement between states, both forward and backward. When you add a custom state, the system automatically includes transitions to all other inherited states, except for the Removed state.

Each state is categorized to support Agile tool backlogs and board views, ensuring a streamlined and organized workflow process.

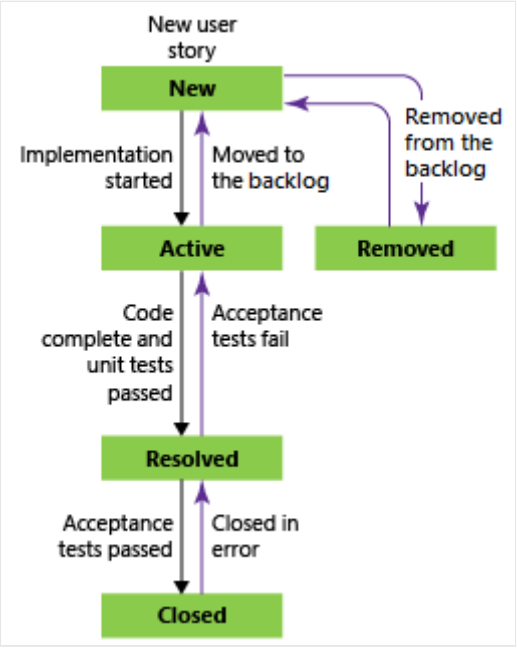
Workflow states

Workflow states define how a work item progresses from creation to closure. For the User Story (Agile process), the main states are New, Active, Resolved, and Closed. The Removed state is used to remove a work item from the backlog; for more information, see Move, change, or delete work items.

The natural progressions and regressions for work item types—user story (Agile), issue (Basic), product backlog item (Scrum), and requirement (CMMI)—are as shown.

Agile process

Workflow states: User Story, Agile process



Category states

Category states determine how Agile planning tools and dashboard widgets treat each workflow state. Work item types use state categories to track progress. These states apply across all projects using the same process and affect how work items appear on backlogs and boards. The state categories used by backlogs, boards, and widgets are *Proposed*, *In Progress*, *Resolved*, and *Complete*.

The following table shows how the default, inherited states map to the category states for the four system processes, including Test Plan work item types. The workflow states for Test Case, Test Design, and Test Suite are the same across all four system processes.

Agile process

Categories

Work tracking

Test tracking

Proposed: Assigned to states associated with newly added work items so that they appear on the backlog. The first column on the boards and Taskboards map to a Proposed state category.

New

Design (Test Case)

In Progress: Assigned to states that represent active work. Work items assigned to states mapped to this category appear in the backlog (unless you choose to hide them) and make up the middle columns on boards.

Active (Bug, Epic, Feature, User Story)

Active (Test Plan) In Planning (Test Suite) In Progress (Test Suite) Ready (Test Case)

Resolved: Assigned to states that represent a solution was implemented, but not yet verified. Generally these states apply to bugs. Work items in a *Resolved* category state appear on the backlog by default. You can also include *Resolved* states in burndown charts, providing a more accurate tracking of progress. The Agile tools treat the *Resolved* category state exactly the same as the *In Progress* category state.

Resolved (Bug)

n/a

Completed: Assigned to states that represent work that's complete. Work items whose state is in this category don't appear on the backlog and do appear in the last column of the board. You can't modify states in this category nor can you add states to this category.

Closed (Bug, Epic, Feature, User Story)

Closed (Test Case) Completed (Test Suite) Inactive (Test Plan)

Removed: Assigned to the Removed state. Work items in a state mapped to the Removed category are hidden from the backlog and board experiences.

Removed (Epic, Feature, User Story)

n/a

Work item types and their boards

Understand where each work item type appears so you can manage your work effectively.

Expand table

Work item type category	Work items appear here
Requirement	Only on the product board.
Feature	Only on the Feature portfolio board.
Epic	Only on the Epic portfolio board.
Custom	Only on a custom portfolio board.

Tip

We recommend that you map each workflow state to a column. If it's not mapped, it doesn't appear on the board.

Note

Completed or closed work items don't display on the backlogs and boards after their **Changed Date** value is greater than 183 days (about a half a year). You can still list these items by using a query. If you want them to show up on a backlog or board, you can make a minor change to them, which resets the clock.

Activated By/Date and Resolved By/Date fields

The system updates these fields—**Activated By**, **Activated Date**, **Resolved By**, and **Resolved Date**—when a change occurs based on corresponding workflow category states. When the workflow state changes to an *In Progress* state category, **Activated By** and **Activated Date** are updated. When the workflow state changes to a *Resolved* state category, **Resolved By** and **Resolved Date** are updated.

To learn more how workflow states map to state categories, see How workflow states and state categories are used in Backlogs and Boards.

Note

The logic governing the fields described here applies to Azure DevOps Services, Azure DevOps Server 2020.1 update, and later versions.

Because these fields reference the workflow state categories, custom workflow states that you add are referenced when updating the fields. To learn more about customization, see Customize the workflow for a process.

Additional notes:

- The fields get updated anytime a work item moves from any category state other than that being set. For example, if you update a work item from *New* to *Fixed*, the **Resolved By/Resolved Date** fields are updated. However, if you update from *Fixed* and *Ready for Testing*—which are in the same category state—the **Resolved By/Resolved Date** fields aren't updated.
- When you transition backwards, such as going from a *Resolved* to an *Active* state, the system clears the values for **Resolved By/Resolved Date** fields. If you got from *Active* to *New*, the system clears the values for **Activated By/Activated Date** fields.
- Don't manually change the values for these fields. They are system fields that are governed by system rules. Any value you attempt to set gets over written.

When to add a State versus a column

Use both States and columns to track work status. Workflow states are shared across a project, while columns are shared within a team. Only project collection admins can add custom states, while team admins can add columns.

Add custom states to align all teams with the organization's business workflow. Customizing the process automatically customizes the projects and work item types that reference it.

Custom states help avoid confusion from different teams creating queries based on columns. Since teams can customize board columns and swimlanes, work item values might differ across boards. Maintain single ownership of work items by team area path or formalize columns by adding custom states shared across teams.

Auto completion of work items with pull requests

When you link a work item to a pull request (PR), you can automatically complete those work items when you complete the PR. For more information, see Auto complete work items with pull requests.

Automate work item state transitions

You can automatically update the state of a work item according to the state of its child tasks. For more information, see Automate work item state transitions.

Related articles

Inheritance process model

- Customize your workflow
- Apply rules to workflow states
- Evaluate rules
- Explore custom rule scenarios

Dashboard widgets

- Explore Lead Time and Cycle Time control charts (widgets) ::: moniker-end

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Feedback

Was this page helpful? Yes No

Provide product feedback

Reference guide for link types

Article • 03/25/2024

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019

You can use different link types to manage the various relationships between work items and other artifacts, like builds, commits, pull requests, and more. In this article, we describe the following link types.

- **Work link type**: Links work items including select test case management work items
- **Hyperlink**: Connects a work item to any URL or network share
- **External link type**: Connects a work item to an external object, such as a code object, build, or wiki page
- **Remote work link type**: Connects work items that are defined in different organizations
- **GitHub link type**: Connects a work item to a GitHub repository commit, issue, or pull request.

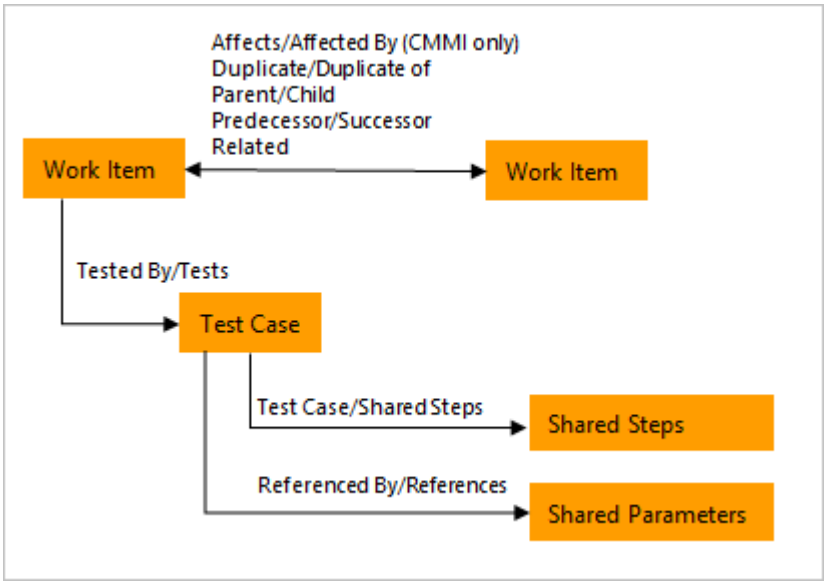
A specific field maintains a count of links for the first four link types, such as *Related Link Count*, *Hyperlink Count*, *External Link Count*, and *Remote Link Count*.

Link types are subject to certain restrictions based on their topology. Use the guidance provided in the following tables to choose which link type to use based on the types of queries and reports you want to create. To learn more about the different topologies, see Link type topologies and restrictions.

Work link type

Work link types are system-defined, process-defined, or user-defined (custom). The links listed in the following table are system-defined.

Each work link type defines the link labels, topology type, and restrictions that are used when links between work items are constructed. For example, the parent-child link type defines two labels: Parent and Child. The link type also supports a hierarchical or tree topology, and prevents circular references from being created between work items.

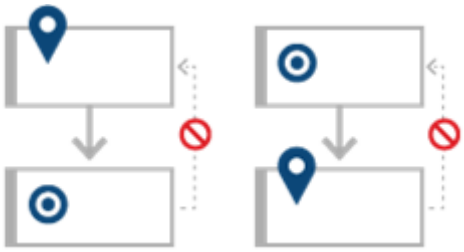


A work item's *Related Link Count* corresponds to the sum of all links defined with a work link type.

Name

Reference name

Usage



Affects-Affected by (CMMI only)

Microsoft.VSTS.Common.Affects-Forward
Microsoft.VSTS.Common.Affects-Reverse
Topology type: Dependency
Link category: Process-defined

Use this directional link to create links between any set of work items, but not ones that would create closed loops. Typically used to track change requests made to requirements.

Restrictions and recommendations:

- You can link a change request to only one requirement using Affects. You can link requirements to as many child change requests as needed using Affected by.
- Only use Affects-Affected by links to link work items in the same project. We recommend this action if you plan to use Excel to modify or update work item data.

Child-Parent



System.LinkTypes.Hierarchy-Forward
System.LinkTypes.Hierarchy-Reverse
Topology type: Tree
Link category: System-defined

Use this directional link to create one-to-many relationships between a single parent to one or more child items. Use to organize work item within a hierarchy. You can quickly create this hierarchy among backlog items using the mapping function or among backlog items and tasks using the sprint backlog or Taskboard.

Typical uses include:

- Maintain task summary relationships. Parent-child links are created for summary tasks and their subordinate tasks.
- Link tasks to PBIs, user stories, or requirements. Supports Backlog Overview, Stories Overview, and Requirements Overview reports.

Restrictions and recommendations:

- Use Excel to bulk edit both work items and parent-child links. See Bulk add or modify work items with Excel.
- A work item can have only one Parent. A parent work item can have many children.
- Only use parent-child links to link work items in the same project. This action is recommended if you plan to use Excel to modify or update work item data.

Duplicate-Duplicate of



System.LinkTypes.Duplicate-Forward
System.LinkTypes.Duplicate-Reverse

Topology type: Tree
Link category: System-defined

Use this directional link to create one-to-many relationships between a single parent to one or more child items. Use to track tasks, bugs, or other work items that are duplicates of one another.

Restrictions and recommendations:

- A work item can have only one Duplicate.
- Only use Duplicate or Duplicate Of links to link work items in the same project. This action is recommended if you plan to use Excel to modify or update work item data.

Referenced By-References



Microsoft.VSTS.TestCase.
SharedParameterReferencedBy
Topology type: Dependency
Link category: Process-defined

Use to link test cases to shared parameters. Use to link Test Cases to Shared Parameters to support the ability to repeat a test with different data. In general, you wouldn't add this link type to a scoped links control.

Related

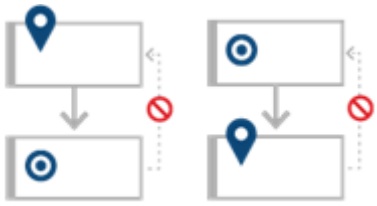


System.LinkTypes.Related
Topology type: Network
Link category: System-defined

Use this nondirectional link to create links between any set of work items. Use to link work items that are at the same level, such as two user stories that define features that overlap one another. The Related link type creates simple relationships with few restrictions.

- Relate work items that are at the same level, such as two user stories that define features that overlap one another.
- Link work items that are defined in different projects and managed by different teams.
- Find and view work items and their related work items in a two-tiered view.
- Create simple relationships with few restrictions.

Successor-Predecessor



System.LinkTypes.Dependency

Topology type: Dependency
Link category: System-defined
Choose **Predecessor** link type when linking to a work item that should be completed before the work item you're linking from. Choose **Successor** link type when linking to a work item that should be completed *after* to the work item you're linking from.

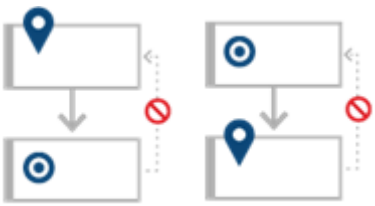
Use this directional link to create links between any set of work items, but not ones that would create closed loops. Use to track tasks that must be completed before others can be started. Linked tasks are represented as predecessor-successor links in Azure Boards.

- Track tasks that must be completed before others can be started.
- Supports one-to-many relationships.
- Find and view predecessor work items and their successor work items in a two-tiered, direct links query view.

Restrictions and recommendations:

- An error appears when you attempt to create links that define circular relationships.
- Create predecessor-successor links only to work items that are within the same project. You can create predecessor-successor links between work items that are defined in different projects. But, if you export a query to Excel, only those work items defined for the project for which the query is defined get imported.

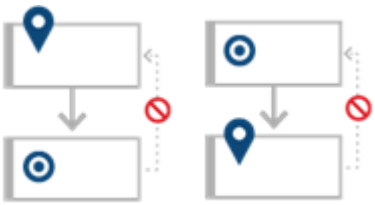
Tested by-Tests



Microsoft.VSTS.Common.TestedBy-Forward
Microsoft.VSTS.Common.TestedBy-Reverse
Topology type: Dependency
Link category: Process-defined

Link test cases to work items, such as bugs, user stories, requirements, and product backlog items. Use to track test cases that test user stories (Agile), product backlog items (Scrum), or requirements (CMMI). Can also link to other work item types such as bugs, issues, or tasks. For on-premises Azure DevOps, there are several SQL reports that depend on these links. See Review team activities to support useful reports.

Test Case-Shared Steps

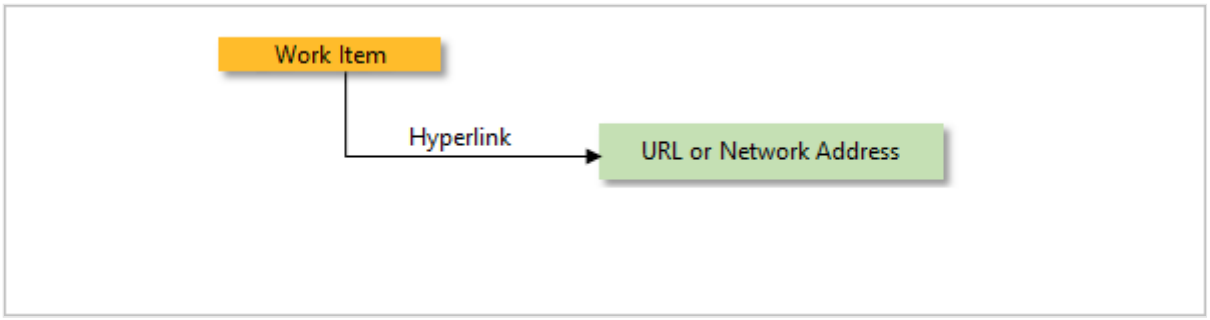


Microsoft.VSTS.TestCase.
SharedStepReferencedBy
Topology type: Dependency
Link category: Process-defined

Use to link test cases with shared steps. Share steps between test cases to avoid having to create multiple entries of the same sequence of steps.

Hyperlink

A hyperlink can link a work item to any URL. The *Hyperlink Count* tracks the number of hyperlinks to a work item.



Link name

Tool supported

Artifact type

Usage

Hyperlink

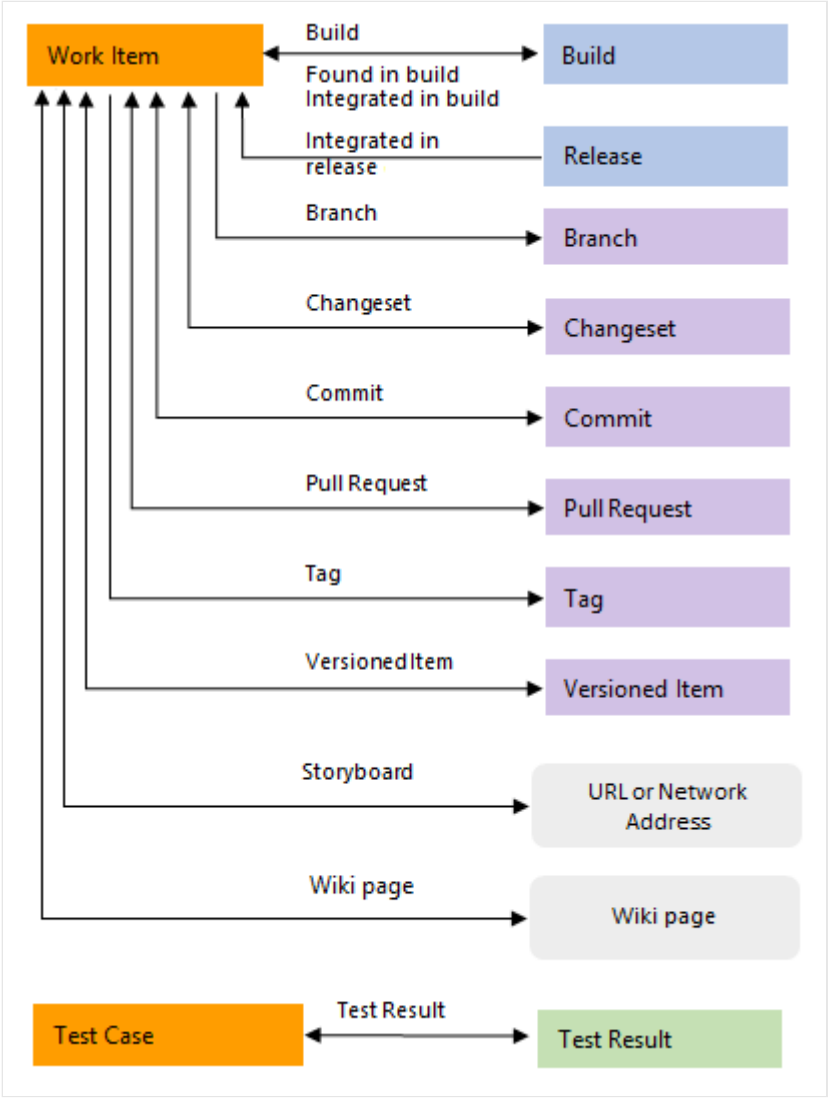
Work item tracking

Hyperlink

Used to link a work item to a URL. **Work item Hyperlink** is the name of this link type in the Artifact Link Types API.

External link type

External link types are system-defined link types that support linking work items to other objects stored within Azure DevOps as shown in the following image. A work item's *External Link Count* corresponds to the sum of all links defined with an external link type.



> ⓘ Note
>
> You can only use an external link type to link to an Azure DevOps object. To link work items to other objects outside of Azure DevOps, use a hyperlink.

The following table describes the external link types you can choose when you add a link type from a work item or test case.

Link name

Tool supported

Artifact type

Usage

Branch

Git

Branch

Used to link a work item to a branch.

Pipelines/Build

Build

Build

Used to link a work item to a build.

Changeset (or Fixed in Changeset)

VersionControl

Changeset

Used to link a work item to a changeset.

Commit (or Fixed in Commit)

Git

Commit

Used to link a work item to a commit.

Found in build

Pipelines/Build

Build

Used to link a work item to a build.

Integrated in build

Build

Build pipeline

Used to link a work item to a build.

Integrated in release environment

Release

Release pipeline

Used to link a release to a work item. The system creates a link of this type when a user enables the **Report deployment status to Work** option for a release definition.

Pull Request

Git

PullRequestId

Used to link a work item to a pull request.

Result attachment

Test Management

TcmResultAttachment

Used to link a work item to an attachment associated with a test result. These links appear when you associate a work item with a test result from **Test** or Microsoft Test Manager.

Source Code File<

VersionControl

LatestItemVersion

Used to link a work item to a file under Team Foundation version control (TFVC).

Storyboard

Requirements

Storyboard

Used to link a work item to a PowerPoint file or other file that contains story boarding information on a network.

Tag

Git

Tag

Used to link a work item to a tag defined for a git commit or git repository. For more information, see Work from the Git command prompt.

Test Result

Test Management

TcmResult

Used to link a work item to a test result. These links appear when you associate a work item with a test result from **Test** or Microsoft Test Manager.

Versioned item

VersionControl

LatestItemVersion

Used to link a work item to a file or changeset defined within a TFVC repository. **Source Code File** is the name of this link type in the Artifact Link Types API.

Wiki

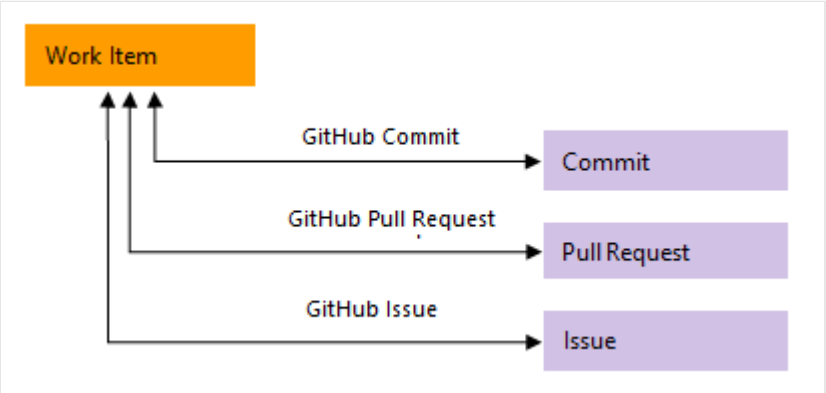
Wiki

Wiki

Used to link a work item to a wiki page. Supported for TFS 2018.2 and later versions.

GitHub link type

GitHub link types are system-defined and support linking work items to GitHub objects as shown in the following image.



> **Important**
>
> You can only link to GitHub artifacts with repositories connected to Azure Boards.

The following table describes the GitHub link types you can choose when adding a link type from a work item.

Link name	Artifact type	Usage
GitHub Commit	GitHub repository commit	Used to link a work item to a GitHub commit.
GitHub Issue	GitHub repository issue	Used to link a work item to a GitHub issue.
GitHub Pull Request	GitHub repository pull request	Used to link a work item to a GitHub pull request.

Remote work link type

Remote work link types are system-defined link types that support linking work items defined in different organizations, as long as the same Microsoft Entra ID manages them as described in Connect your organization to Microsoft Entra ID.

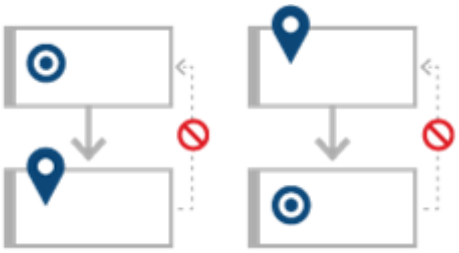
A work item's *Remote Link Count* corresponds to the sum of all links defined with a remote work link type.

Name

Reference name

Usage

Consumes From-Produced For



(Dependency topology)

System.LinkTypes.Remote.Dependency-Forward
System.LinkTypes.Remote.Dependency-Reverse
Topology type: Dependency
Link category: System-defined

Use this directional link to create links between work items that have dependencies and are defined in different organizations, as long as the same Microsoft Entra ID manages them. Typically used to track change requests made to requirements.

Remote Related



System.LinkTypes.Remote.Related
Topology type: Network
Link category: System-defined

Use this nondirectional link to create links between work items defined in different organizations, as long as the same Microsoft Entra ID manage them.

List link types

To get a list of link types, you can use one of the supported command-line tools.

az boards work-item relation list-type

You can list link types supported by your organization with the az boards work-item relation list-type command or the Work Item Relation Types - List REST API command. To get started, see Get started with Azure DevOps CLI.

Azure CLI

```
az boards work-item relation list-type [--org]
```

Optional parameters

- **org**: Azure DevOps organization URL. You can configure the default organization using `az devops configure -d organization=ORG_URL`. Required if not configured as default or picked up using `git config`. Example: `--org https://dev.azure.com/MyOrganizationName/`.

Example

The following command lists the work item link types in table format that are defined for the fabrikam organization.

Azure CLI

```
az boards work-item relation list-type --org fabrikam --output table
Name                  ReferenceName
Enabled    Usage
--------------------  ------------------------------------------------------
-----  ---------  ------------
Produces For          System.LinkTypes.Remote.Dependency-Forward
True       workItemLink
Consumes From         System.LinkTypes.Remote.Dependency-Reverse
True       workItemLink
Duplicate             System.LinkTypes.Duplicate-Forward
True       workItemLink
Duplicate Of          System.LinkTypes.Duplicate-Reverse
True       workItemLink
Referenced By         Microsoft.VSTS.TestCase.SharedParameterReferencedBy-
Forward  True       workItemLink
References            Microsoft.VSTS.TestCase.SharedParameterReferencedBy-
Reverse  True       workItemLink
```

```
Tested By             Microsoft.VSTS.Common.TestedBy-Forward
True       workItemLink
Tests                 Microsoft.VSTS.Common.TestedBy-Reverse
True       workItemLink
Test Case             Microsoft.VSTS.TestCase.SharedStepReferencedBy-Forward
True       workItemLink
Shared Steps          Microsoft.VSTS.TestCase.SharedStepReferencedBy-Reverse
True       workItemLink
Successor             System.LinkTypes.Dependency-Forward
True       workItemLink
Predecessor           System.LinkTypes.Dependency-Reverse
True       workItemLink
Child                 System.LinkTypes.Hierarchy-Forward
True       workItemLink
Parent                System.LinkTypes.Hierarchy-Reverse
True       workItemLink
Related               System.LinkTypes.Related
True       workItemLink
Remote Related        System.LinkTypes.Remote.Related
True       workItemLink
Attached File         AttachedFile
True       resourceLink
Hyperlink             Hyperlink
True       resourceLink
Artifact Link         ArtifactLink
True       resourceLink
```

The default json format provides additional information about the attributes defined for the link types. For example, the information for the link types *Produces For* and *Consumes From* are listed as follows.

Output


```
  {
    "attributes": {
      "acyclic": true,
      "directional": true,
      "editable": false,
      "enabled": true,
      "isForward": true,
      "oppositeEndReferenceName": "System.LinkTypes.Remote.Dependency-
Reverse",
      "remote": true,
      "singleTarget": true,
      "topology": "dependency",
      "usage": "workItemLink"
    },
    "name": "Produces For",
    "referenceName": "System.LinkTypes.Remote.Dependency-Forward",
    "url":
"https://dev.azure.com/mseng/_apis/wit/workItemRelationTypes/System.LinkType
s.Remote.Dependency-Forward"
```

```
  },
  {
    "attributes": {
      "acyclic": true,
      "directional": true,
      "editable": false,
      "enabled": true,
      "isForward": false,
      "oppositeEndReferenceName": "System.LinkTypes.Remote.Dependency-
Forward",
      "remote": true,
      "singleTarget": true,
      "topology": "dependency",
      "usage": "workItemLink"
    },
    "name": "Consumes From",
    "referenceName": "System.LinkTypes.Remote.Dependency-Reverse",
    "url":
"https://dev.azure.com/mseng/_apis/wit/workItemRelationTypes/System.LinkType
s.Remote.Dependency-Reverse"
  },
```

`witadmin listlinktypes`

You can list link types supported for your project collection using the witadmin listlinktypes command-line tool or the Work Item Relation Types - List REST API command.

Here we list the link types for the fabrikam-sever default collection:

CLI

```
C:\Program Files (x86)\Microsoft Visual
Studio\2019\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\
Team Explorer>witadmin listlinktypes /collection:http://fabrikam-
server/DefaultCollection

Reference Name: Microsoft.VSTS.TestCase.SharedParameterReferencedBy
Names: Referenced By, References
Topology: Dependency
Is Active: True

Reference Name: Microsoft.VSTS.Common.TestedBy
Names: Tested By, Tests
Topology: Dependency
Is Active: True

Reference Name: Microsoft.VSTS.TestCase.SharedStepReferencedBy
Names: Test Case, Shared Steps
Topology: Dependency
```

```
Is Active: True

Reference Name: System.LinkTypes.Duplicate
Names: Duplicate, Duplicate Of
Topology: Tree
Is Active: True

Reference Name: System.LinkTypes.Dependency
Names: Successor, Predecessor
Topology: Dependency
Is Active: True

Reference Name: System.LinkTypes.Hierarchy
Names: Child, Parent
Topology: Tree
Is Active: True

Reference Name: System.LinkTypes.Related
Name: Related
Topology: Network
Is Active: True
```

Link type attributes

The following table provides descriptions for each of the link type attributes returned by **Azure Boards** CLI or the REST API.

Attribute

Description

Names, `name`

Specifies the friendly name assigned to the link type(s). Directional links are defined in pairs, therefore include a forward and reverse name.

Reference name, `referenceName`

Specifies the name assigned to the link type or link type pair.

`acyclic`

Indicates whether the link type allows or (`true`) or restricts (`false`) circular relationships. For example, tree type links restrict circular relationships. For more information, see LinkTypes elements reference.

`directional`

Indicates whether the link type is directional (`true`) or not (`false`). Directional link types get defined in pairs with a forward and reverse component. For more information, see LinkTypes elements reference.

`editable`

Indicates whether the link type is editable (`true`) or not (`false`). You can only add and edit custom link types for on-premises deployments using witadmin Manage link type command-line tool. System link types always have `editable=false`.

Is Active, `enabled`

Indicates whether the link type is active (`true`) or not (`false`). You can only use custom link types for on-premises deployments using the witadmin Manage link type command-line tool.

`isForward`

Indicates whether the link type specifies the forward link (`true`) or not (`False`) within a link type pair.

`oppositeEndReferenceName`

Specifies the reference name of the link type that defines the link in the opposite direction of a link type pair.

`remote`

Indicates whether the link type supports linking to a remote work item (`true`) or not (`False`). Link types with `remote=false` require that the target work item resides in the same organization or collection as the origin work item.

`singleTarget`

Indicates whether the link type allows for more than one target (`false`) or is restricted to a single target (`true`).

`topology`

Specifies the topology type—dependency`, `network`, and `tree`. For descriptions, see Link type topologies and restrictions.

`usage`

Specifies the usage type—resourceLink or `workItemLink`. The workItemLink`value indicates a link type that links two work items. The`resourceLink` value indicates a link type used to link a work item to a resource, such as a URL or attachment.

`url`

Lists the attributes of the link type in json format.

Related articles

- Link work items to other objects
- Track dependencies using Delivery Plans
- Use mapping to link backlog items to features and epics
- Bulk modify links using Excel
- Artifact Link Types API

Feedback

Was this page helpful? Yes No

Provide product feedback

Classification Nodes

Reference

Service: Work Item Tracking

API Version: 7.1

Operations

Expand table

Create Or Update	Create new or update an existing classification node.
Delete	Delete an existing classification node.
Get	Gets the classification node for a given node path.
Get Classification Nodes	Gets root classification nodes or list of classification nodes for a given list of nodes ids, for a given project. In case ids parameter is supplied you will g...
Get Root Nodes	Gets root classification nodes under the project.
Update	Update an existing classification node.

Processes - List

Reference

Service: Work Item Tracking Process

API Version: 7.1

Get list of all processes including system and inherited.

HTTP

```
GET https://dev.azure.com/{organization}/_apis/work/processes?api-version=7.1
```

With optional parameters:

HTTP

```
GET https://dev.azure.com/{organization}/_apis/work/processes?$expand={$expand}&api-version=7.1
```

URI Parameters

Expand table

Name	In	Required	Type	Description
organization	path	True	string	The name of the Azure DevOps organization.
api-version	query	True	string	Version of the API to use. This should be set to '7.1' to use this version of the api.
$expand	query		GetProcess ExpandLevel	

Responses

Expand table

Name	Type	Description
200 OK	Process Info[]	successful operation

Security

oauth2

Type: oauth2
Flow: accessCode
Authorization URL: https://app.vssps.visualstudio.com/oauth2/authorize&response_type=Assertion
Token URL: https://app.vssps.visualstudio.com/oauth2/token?client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer

Scopes

Expand table

Name	Description
vso.work	Grants the ability to read work items, queries, boards, area and iterations paths, and other work item tracking related metadata. Also grants the ability to execute queries, search work items and to receive notifications about work item events via service hooks.

Examples

Get the list of processes

Sample request

HTTP

HTTP

```
GET https://dev.azure.com/{organization}/_apis/work/processes?api-version=7.1
```

Sample response

Status code: 200

JSON


```
{
  "count": 14,
  "value": [
    {
      "typeId": "adcc42ab-9882-485e-a3ed-7678f01f66bc",
      "referenceName": null,
      "name": "Agile",
      "description": "This template is flexible and will work great for most
teams using Agile planning methods, including those practicing Scrum.",
      "parentProcessTypeId": "00000000-0000-0000-0000-000000000000",
      "isEnabled": true,
      "isDefault": true,
      "customizationType": "System"
    },
    {
      "typeId": "27450541-8e31-4150-9947-dc59f998fc01",
      "referenceName": null,
      "name": "CMMI",
      "description": "This template is for more formal projects requiring a
framework for process improvement and an auditable record of decisions.",
      "parentProcessTypeId": "00000000-0000-0000-0000-000000000000",
      "isEnabled": true,
      "isDefault": false,
      "customizationType": "System"
    },
    {
      "typeId": "6b724908-ef14-45cf-84f8-768b5384da45",
      "referenceName": null,
      "name": "Scrum",
      "description": "This template is for teams who follow the Scrum
framework.",
      "parentProcessTypeId": "00000000-0000-0000-0000-000000000000",
      "isEnabled": true,
      "isDefault": false,
      "customizationType": "System"
    },
    {
      "typeId": "e31fd6e1-c6af-44bb-925e-dfa71f551463",
      "referenceName": null,
      "name": "CustomProcess",
      "description": "This template is flexible and will work great for most
teams using Agile planning methods, including those practicing Scrum.",
      "parentProcessTypeId": "00000000-0000-0000-0000-000000000000",
      "isEnabled": true,
      "isDefault": false,
      "customizationType": "Inherited"
    }
  ]
}
```

Definitions

Expand table

Name	Description
CustomizationType	Indicates the type of customization on this work item. System behaviors are inherited from parent process but not modified. Inherited behaviors are modified behaviors that were inherited from parent process. Custom behaviors are behaviors created by user in current process.
GetProcessExpandLevel	
ProcessInfo	Process.
ProjectReference	Defines the project reference class.

CustomizationType

Indicates the type of customization on this work item. System behaviors are inherited from parent process but not modified. Inherited behaviors are modified behaviors that were inherited from parent process. Custom behaviors are behaviors created by user in current process.

Expand table

Name	Type	Description
custom	string	Customization-type is Custom if the newly created workitemtype is customized.
inherited	string	Customization-type is Inherited if the existing workitemtype of inherited process is customized.
system	string	Customization-type is System if is system generated workitemtype.

GetProcessExpandLevel

Expand table

Name	Type	Description
none	string	No expand level.

projects	string	Projects expand level.

ProcessInfo

Process.

Expand table

Name	Type	Description
customizationType	Customization Type	Indicates the type of customization on this process. System Process is default process. Inherited Process is modified process that was System process before.
description	string	Description of the process.
isDefault	boolean	Is the process default.
isEnabled	boolean	Is the process enabled.
name	string	Name of the process.
parentProcessTypeId	string	ID of the parent process.
projects	Project Reference[]	Projects in this process to which the user is subscribed to.
referenceName	string	Reference name of the process.
typeId	string	The ID of the process.

ProjectReference

Defines the project reference class.

Expand table

Name	Type	Description
description	string	Description of the project
id	string	The ID of the project
name	string	Name of the project
url	string	Url of the project

Projects

Reference

Service: Core

API Version: 7.1

DO NOT REMOVE. Used in TfsDeleteProject.exe to determine whether soft-delete is enabled.

Operations

Expand table

Create	Queues a project to be created. Use the GetOperation to periodically check for create project status.
Delete	Queues a project to be deleted. Use the GetOperation to periodically check for delete project status.
Get	Get project with the specified id or name, optionally including capabilities.
Get Project Properties	Get a collection of team project properties.
List	Get all projects in the organization that the authenticated user has access to.
Set Project Properties	Create, update, and delete team project properties.
Update	Update an existing project's name, abbreviation, description, or restore a project.

Teams

Reference

Service: Core

API Version: 5.0

Manage teams in Azure DevOps Services. A DevOps account has one or more team projects. Each team project has one or more teams that contribute to that project.

Operations

Expand table

Create	Create a team in a team project. Possible failure scenarios Invalid project name/ID (project doesn't exist) 404 Invalid team name or description 400 Team already ex...
Delete	Delete a team.
Get	Get a specific team.
Get All Teams	Get a list of all teams.
Get Team Members With Extended Properties	Get a list of members for a specific team.
Get Teams	Get a list of teams.
Update	Update a team's name and/or description.

Azure Boards documentation

Plan, track, and discuss work across teams. Define and update issues, bugs, user stories, & other work with customizable Scrum, Kanban, and Agile tools.

About Azure Boards

OVERVIEW

What is Azure Boards?

About work items

CONCEPT

About default processes

Agile process workflow

Best practices for Agile project management

Get started

QUICKSTART

Sign up for Azure Boards

Plan & track work

Get started as a Stakeholder

Plan your project

OVERVIEW

About Backlogs & Agile project management

QUICKSTART

Create your backlog

Define features & epics

Organize your backlog

TUTORIAL

Bulk add or modify work items (Excel)

Bulk add or modify work items (Web)

Implement Kanban

QUICKSTART

Board quickstart

TUTORIAL

Board overview

Add columns

Customize your board

Implement Scrum

OVERVIEW

About Sprints & Scrum

TUTORIAL

Assign work to sprints

Set capacity

Monitor sprint burndown

List & manage work items

TUTORIAL

Create managed queries

Do ad hoc searches

Define, triage, & manage bugs

Remove, delete, or restore work items

Collaborate

QUICKSTART

Connect Azure Boards to GitHub

Link GitHub commits, pull requests, & branches to work items

CONCEPT

Link work items to other objects

TUTORIAL

Use work item templates

Add tags to work items

Azure Boards with Microsoft Teams

Configure & customize

CONCEPT

About teams & Agile tools

Configure & customize Boards

TUTORIAL

Set area paths

Set iteration (sprint) paths

Add a custom field

Customize the workflow

See more...

Agile at scale

TUTORIAL

Portfolio management

Review team deliverables & plans

Implement Scaled Agile Framework®

Troubleshoot

HOW-TO GUIDE

Fix reordering & nesting issues

Set up your Backlogs & Boards

FAQs

Quick reference

REFERENCE

Work items quick reference

Query quick reference

Work item field index

Developer resources

REFERENCE

Azure DevOps CLI

REST API

Settings & Usage documentation

Configure resources and manage settings for an organization, project, team, or user.

Get started

OVERVIEW

About settings

Get started as an administrator

QUICKSTART

Set user preferences

HOW-TO GUIDE

Enable preview features

Manage your organization (cloud)

OVERVIEW

About managing your organization

Plan your organizational structure

CONCEPT

Manage access with Microsoft Entra ID

QUICKSTART

Create an organization

HOW-TO GUIDE

Add users to your organization

Connect your organization to Microsoft Entra ID

Add & manage projects

CONCEPT

About projects & scaling up

Customize your project

OVERVIEW

Manage your project

HOW-TO GUIDE

Create a project

Connect to GitHub

Add & manage teams

CONCEPT

About teams & Agile tools

HOW-TO GUIDE

Add a team

Configure team tools

Define team area paths

Configure team iterations

Add a team administrator

Set alerts or notifications

CONCEPT

About notifications

HOW-TO GUIDE

Set personal notifications

Set team notifications

Audit & usage (cloud)

CONCEPT

Rate limits & usage

QUICKSTART

Access, export, and filter audit logs

Configure Pipelines resources

HOW-TO GUIDE

Create & manage agent pools

Create & manage deployment groups

Set build & release retention policies

Create & manage service connections

Configure Repos resources

OVERVIEW

About branches and branch policies

HOW-TO GUIDE

Create & manage Git repositories

Set Git repository settings and policies

Manage Git branch policies

Add TFVC check-in policies

Configure Test resources

HOW-TO GUIDE

Set test retention policies

Extensions overview

Article • 11/28/2023

Azure DevOps Services | Azure DevOps Server 2022 - Azure DevOps Server 2019 | TFS 2018

Extensions are add-ons, which you can use to customize and extend your experience with Azure DevOps. They're written with standard technologies - HTML, JavaScript, CSS - and developed using your preferred development tools.

Extensions get published at the Visual Studio Marketplace, where they can be kept private for you and your team or shared publicly with the millions of developers currently using Azure DevOps.

Extensions use our RESTful API Library to easily interact with Azure DevOps and applications/services.

Understand parts of an extension



The following items make up an extension:

- JSON manifest file: Contains basic info about the extension.
- Discovery assets: Markdown and images that make up the extension's overview and aesthetics in the Marketplace.
- Static files: Contain the logic of the extension, including HTML, JS, and CSS files. Static files are only applicable to contribution-based extensions.

These files and assets get bundled up to make a VSIX file that gets published to the Marketplace.

From the Marketplace, users can install extensions directly into their organization. If you don't have permissions to install an extension, but you're a project member, you can

request an extension instead.

Use an extension

There are dozens of ways you can use an extension and places where you can add to the user interface, and we're adding more every sprint. Learn about all of the places where you can add a hub in the Extensibility points.

- Provide new Azure Pipelines tasks that teams can use in their builds.
- Use dashboard widgets to get custom views within Azure DevOps.
- Extend the work item form with new tabs, sections, and actions.
- Create your own hub to embed new capabilities within our Agile, code, build, and test experiences.
- Develop actions that can be run on hubs, whether they're ours or ones you created.

Evaluate a Marketplace extension

To evaluate a Marketplace extension, review the information and resources described in the following table. You can find this information in the extension information

Information

Usage

Top Publisher badge



The publisher demonstrates commitment to its customers and the Marketplace through excellent policies, quality, reliability, and support. For more information, see Top Publisher.

Q & A

The Q & A section of published extensions might answer questions you have. Also, they're a good mechanism to engage with the extension's publisher(s) to have a meaningful dialogue to make yourself comfortable. Use the Q & A information to

understand the development, testing, and security practices the publisher follows. It also gives you a sense of the publisher's responsiveness.

Ratings & reviews

Ratings and reviews indicate how others perceive the offering. For more information, see Respond to customer feedback.

Privacy, license, and support policies

See if the publisher provided them and if they meet your needs or concerns. For more information, go to Safety information.

Safety information

- **Malware scan**: The Marketplace runs a virus scan on each new and updated extension package that publishes to ensure its safety. Until the scan is all clear, we don't publish the extension for public usage. If a concern surfaces, the Marketplace team has the means to disable an extension immediately and notify its existing customers.
- **Content scan**: The Marketplace scans content for every new and updated extension to avoid surfacing inappropriate or offensive content on the Marketplace pages.
- **Access to approved scopes only**: An extension can only operate within the granted scopes. For example, an extension that has only read permissions on work items can't modify your features and bugs. Azure DevOps web extensions run in a sandboxed browser IFrame. They're only able to access Azure DevOps data and APIs approved for the extension. During installation, admins are prompted to approve permissions and scopes. One way to protect yourself is to carefully review the scopes the extension requests.

  > **Note**
  >
  > If the scopes change for an extension, the you must approve the update before it can become applicable on your organization or collection.

- **Third-party build and release tasks**: Tasks are implemented as code that executes on an agent machine. Tasks are only able to access secrets explicitly provided to them (see variable secrets), but tasks generally have full access to the agent

machine itself. To reduce risk, run builds on Microsoft-hosted agents, which are VMs isolated from other jobs and recycled after each job. Or, limit file and network access on private hosted agent machines. Learn more about build and release agents.

- **Third-party code execution on the server**: There's no way for an extension to install or execute any code on Azure DevOps Server.

Build an extension

Before you build an extension, familiarize yourself with the extension types already available within the Marketplace, Extensions for Azure DevOps. Learn how to build your first extension and check out our full set samples.

- Develop your first extension
- Samples

For more information about building extensions, see the following articles:

- REST APIs
- Service Hooks
- Package, publish, and install your extension
- Package and publish your integration with an external app or service
- Share your work publicly with the entire community

Next steps

Develop an extension

Related articles

- Visual Studio Marketplace
- Extension Publisher Page
- Visual Studio Partner Program
- Extension manifest reference

Deprecation of the Work Item Tracking (WIT) and Test Client OM

Article • 02/24/2023

Azure DevOps Services

WIT and Test Client OM are part of the broader TFS SDK. They are a set of SOAP-based APIs that can be used to create tools to run operations against the Azure DevOps. These SOAP-based APIs have been replaced by the set of modern REST-based endpoints. Therefore we are in the process of deprecating the SOAP-based APIs (Client OM) for WIT and Test.

> **Note**
>
> WIT and Test Client OM for Azure DevOps has been deprecated in latest version Visual Studio 2019 and the **Microsoft.TeamFoundationServer.ExtendedClient** package. This means, there's no new work item tracking or test functionality included into the Client OM.

Affected object models

Only the WIT and Test Client OM are being deprecated. Other areas like build and source control aren't on a deprecation schedule at this time.

Impact

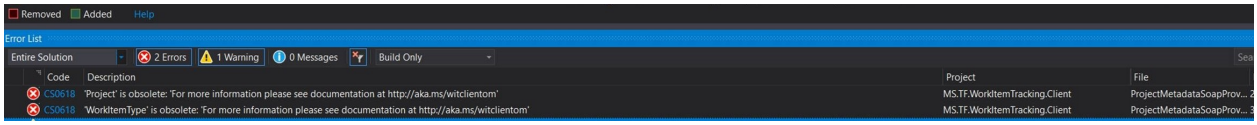
If your organization is using the WIT Client OM in tools that run against Azure DevOps, we recommend you start migrating those tools to stop using the WIT Client OM and start using the new REST based endpoints.

Updating your code to use REST is vital to ensure your tools work as we release new functionality into Azure Boards and Azure Test.

Visual Studio warnings

Any references to the latest versions of WIT Client OM result in an obsolete warning. This includes the dlls that come with Visual Studio 2019 and Azure DevOps 2019. Here is an example warning you may receive...

CS0618: 'WorkItemType' is obsolete: 'for more information, see documentation at https://aka.ms/witclientom '



Q&A

How do I fix my tools?

First thing you should investigate and determine if the tool is still needed. We have found that many organizations create a tool for a specific reason for a short period of time. Then for one reason or another, that tool continues to run when it's no longer needed. For this scenario, you can shut down the tool.

If the tool is still of use, you need to re-refactor that tool using the REST based .NET client libraries. Refactoring requires some reaction work on your part to remove references to the old WIT Client OM and replace them with the appropriate .NET Client Libraries.

We have also put together this handy migration guide with sample code to aid your transition from WIT Client OM to the REST-based .NET Client libraries.

Resources

- Migration Guide
- Azure DevOps Services REST API Reference
- .NET client libraries for Azure DevOps
- Understanding Rate limits
- Integration Best Practices
- Sample Code on GitHub