

1. What's New .....	3
2. User Guide .....	5
2.1 Tasktop Editions .....	6
2.2 Key Concepts .....	8
2.3 User Management .....	22
2.4 Quick Start Guide .....	44
2.4.1 Step 1: Connect to Your Repository .....	
2.4.1.1 Standard Repository Connection .....	51
2.4.1.2 Database Repository Connection .....	60
2.4.2 Step 2: Create or Reuse a Model .....	67
2.4.3 Step 3: Create Your Collection(s) .....	
2.4.3.1 Work Item Collection (Repository) .....	80
2.4.3.1.1 Field Mapping .....	
2.4.3.1.2 Relationship Specification .....	
2.4.3.1.3 Person Reconciliation .....	116
2.4.3.1.4 Comment Configuration .....	
2.4.3.1.5 State Transitions .....	121
2.4.3.1.6 Artifact Unions .....	
2.4.3.2 Container Collection (Repository) .....	
2.4.3.3 Work Item Collection (Database) .....	
2.4.3.4 Gateway Collection .....	
2.4.3.5 Outbound Only Collection .....	
2.4.4 Step 4: Configure your Integration .....	
2.4.4.1 Work Item Synchronization .....	163
2.4.4.1.1 Artifact Creation Flow .....	
2.4.4.1.2 Field Flow .....	171
2.4.4.1.3 Artifact Routing .....	178
2.4.4.1.4 Artifact Filtering .....	
2.4.4.1.5 Comment Flow .....	
2.4.4.1.6 Attachment Flow .....	
2.4.4.1.7 Conflict Resolution .....	
2.4.4.1.8 Change Detection .....	
2.4.4.1.9 Twinless Artifact Update .....	
2.4.4.1.10 Running Your Integration(s) .....	
2.4.4.1.11 Viewing Your Integration(s) .....	
2.4.4.1.12 Tips and Tricks .....	
2.4.4.2 Container + Work Item Synchronization .....	
2.4.4.3 Create via Gateway .....	239
2.4.4.4 Modify via Gateway .....	256
2.4.4.5 Enterprise Data Stream .....	
2.4.4.6 Code Traceability: Create and Relate a Changeset .....	
2.4.4.7 Code Traceability: Update Existing Work Item .....	
2.4.4.8 Test Synchronization .....	316
2.4.5 Step 5: Expand or Modify your Integration .....	
2.5 Resources .....	
2.6 PDF Download .....	361

3. Supported Repository Versions .....	362
3.1 Repository Versions: End-of-Life Dates .....	372

# What's New

## In this release

[Emails en masse](#) | [Save, and save often](#) | [#nofilter](#) | [Asana and Xray On-Prem Are Now Supported](#)

Our big news, in case you haven't heard, is that [Tasktop has joined the Planview family!](#) We will continue to grow and improve our products while providing world-class service. Our Product teams continue to be hard at work, and we wanted to share the latest features, tools, and connectors they've been working on.

This month, we're excited to announce two new connectors—Asana and Xray On-Prem, along with some Hub UI and performance improvements that should make your life a little bit easier.

Read on to learn more.

## Emails en masse

Keeping teams in their tools of choice is kind of what we do, and so, for on-prem customers, we made it possible for you to stay in Hub and send all your email notifications—even when you're sending to multiple addresses.

## Save, and save often

That sinking feeling you get when you do all kinds of work and then forget to save ... it's a thing of the past. You'll notice the "Save" and "Cancel" buttons are now pervasive, even when you scroll down. Never lose data again!

## #nofilter

Filters can be good. For photos, for water, and sometimes, for metrics. But it's important to understand when they're in place, which is why we added some additional information to the Metrics page. We wanted you to know for sure whether the numbers you're looking at—for artifacts and users—have any filters applied.

## Asana and Xray On-Prem Are Now Supported

**Xray** is a popular Jira native application for enterprise test management that streamlines development and testing by managing manual and automated tests as Jira issues. In complex development and test environments, linking development directly to test planning and execution ensures quality is embedded throughout your software development lifecycle.

Synchronizing requirements to build test cases at the beginning of the process improves collaboration between Product and Test. And connecting test execution and results ensures test coverage is thorough and complete. Our new Xray support means testers are included every step of the way and understand the scope of their work for each release at the very beginning. No more silos, no more last-minute work.


You can see Xray in action in the video below or get all the details in our [docs](#).

**Asana** is a web and mobile work management platform designed to help teams organize, track, and manage their work. Our latest connector lets you measure flow in Asana and synchronize artifacts between Asana and the rest of your toolchain.

Watch this demo of an Asana, LeanKit, and Jira integration to see how Product Managers, UX Designers, and Developers can all stay in their preferred tools but still maintain a global view of all departments' progress. Key fields and information about the project's tasks, including section, comments, dates, and attachments will all flow bi-directionally.

If you need to dive deep, you can always check out our [docs](#) to find out more.

# User Guide

 Our product documentation is now available in the [Planview Customer Success Center](#). The documentation will remain available on this site but will no longer be updated. If you have any questions, please reach out to [customer care](#).

## Welcome to User Guide

22.3 Release (July 19, 2022)

**New to Hub? You can:**


- Learn about our product's [key concepts](#)
- Read about [hardware requirements and installation](#)
- Explore our [Quick Start Guide](#)
- Learn about our [Connectors](#)
- Check out our [Release Notes](#)


























 Need help? [Contact support here](#)

# Tasktop Editions


Tasktop Hub is available in **three** editions for on premise and cloud versions — Pro, Enterprise, and Ultimate.

In the table below, you can see which features are included in your edition.

 If you are interested in learning more about other editions, please [contact us](#).

	Pro	Enterprise	Ultimate
<b>Tasktop Viz</b>			
Viz Unlock Package		Available as add-on	Available as add-on
<b>Lifecycle Connectors</b>			
Included Lifecycle Connectors	Connect Any 2 Lifecycle Tools	Connect up to 3 Lifecycle Tools	Unlimited
<b>Automation</b>			
Gateway Integration Style (Create via Gateway Template; Modify via Gateway Template)			
Integration Metrics Dashboard			
Twinless Artifact Update			
Test Synchronization (via Nested Container Integration and Test Step Flow)			
Change Detection (via Cron Expression)			
Key-Value Stores			
Field-by-Field Conflict Resolution			
Conditional Field Flow			

## Visibility

Enterprise Data Stream (Enterprise Data Stream Template)			
Integration Landscape View			
Associated Configuration Elements			
Configuration History			

# Key Concepts

## Overview

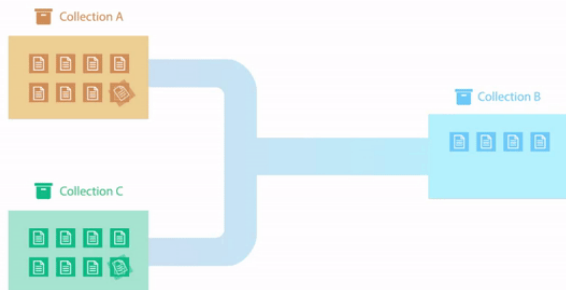
Tasktop is a powerful tool for **connecting your software delivery systems to empower teams, enhance communication, and improve the process of software development as a whole**. Below is a look at some of the concepts Tasktop utilizes to facilitate integration.

The **key concepts** to understand are:

## Integration

At the highest level, the definition of an **integration** is simply the flow of information between **two or more tools**. If you dig a little bit deeper, the definition of an integration is the flow of information, defined by the flow specification, between two or more collections. And collections are sets of artifacts. But that is probably too much to swallow right at the beginning – so don't try to!

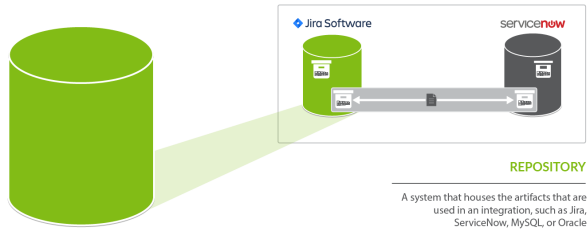
Take a look at a conceptual picture of what an integration looks like in the figure below, and just keep that in mind as we walk through all of the other concepts – then when you come back to this it will make a lot more sense!



So let's first talk about the underpinnings of how Tasktop communicates with end systems, which we call **Repositories**. For all repositories Tasktop connects to, we create what we call a **Repository Connection**. Once we've introduced those concepts we'll talk about **Artifacts** and **Collections** and then we will come back to **Integrations** and talk more about the **flow specification**.

## Repository





A **repository** is **any system that houses the artifacts that can be used in an integration**. Repositories can be systems used as part of the software delivery process, like **Micro Focus (HPE ALM), CA Agile Central, Jira**, etc., or repositories can be more generic databases, like **MySQL** or **Oracle**.

A **repository connection** is a **connection to a specific instance of a given repository that permits Tasktop to communicate with that repository**. To configure a repository connection, users will need to provide base credentials such as a server URL, a username, and a password. You can learn how to set up a repository connection [here](#).

## Artifact



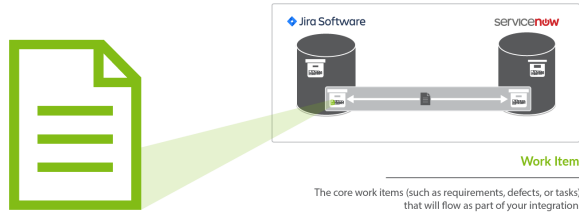
An **artifact** is **any object containing metadata that resides within your repository**. There are two main types of artifacts: **work items** and **containers**. Work items and containers have some similarities, and some key differences, with regard to how they behave within Tasktop Hub.

## Work Item

**Work items** are the **artifacts that are produced by different teams during software development. They are the core items that will flow as part of your integration**. Some examples of common work items are defects, stories, requirements, test cases, and help tickets, to name just a few. Serving as the core currency of communication, work items are the means by which all the work around software production is recorded and tracked. Work items are at the core of any integration and are the entities that Tasktop can create or modify as a part of an integration.

Within Tasktop, you will primarily use work items to:

- Serve as the entity that flows from one repository to the other as part of your integration. For example, you can flow requirements in your source repository to your target repository, where they will create corresponding requirements.



## Container

**Containers** are artifacts that are used to group work items. They define where, within the repository, each work item resides. Some examples of common containers are projects, folders, modules, workspaces, and sets. The main purpose of a container is to define a set of work items.

Within Tasktop, you primarily use containers to:

- Define the scope of your collection. For example, you could add Project A and Project B to your collection, which would mean only artifacts within those projects would be eligible to flow in your integration (we'll explain this more in the **Collection** section).
- Define routing for your collection. Routing defines *where* artifacts will be created within your target collection. For example, if you route Project A in Jira to Project B in Jama, that will tell Tasktop to flow artifacts in Project A in Jira over to Project B in Jama, where they will create corresponding artifacts.
- For specific low-level container types, you can create a [Container Collection](#), which will allow you to flow Containers from a source Collection to a Target Collection — allowing you to recreate your container (i.e., folder, module, component) structure, as well as the work items contained within them in the target repository.

## High-Level Containers vs. Low-Level Containers

Some repositories contain *high level containers*, such as workspaces, which are then broken into *low level containers*, such as projects.

### Container types



Containers are a key component of creating your collection, as each collection is defined by its artifact type (i.e. defect, requirement, test case, etc), by the model it is mapped to, and by the *high level containers* it includes. In this way, containers are essential for how you define which artifacts can flow as part of your integration.

You can learn more about how to select the containers included in your collection [here](#).

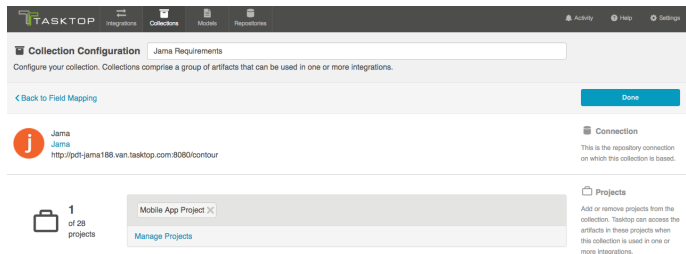
Your containers also become important during the Artifact Routing stage of configuring your integration. On the Artifact Routing Screen, you are able to determine how artifacts should flow from one collection's containers to the other's. Some repositories allow you to route at only the *low level container* level, some allow you to route at the *high level container* level, and others allow a mixed approach.

You can learn more about how to configure artifact routing [here](#).

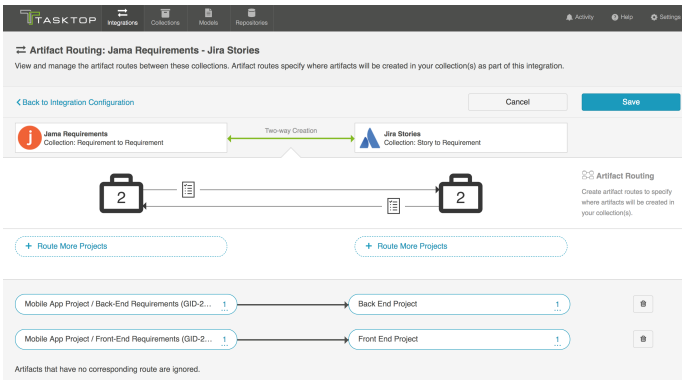
To understand this better, let's look at an example in Jama. Jama contains *high-level containers* (projects) which are then divided into several *low-level containers* (sets), which contain *work items* (requirements, in this case). Here, our *high-level container* is the Mobile App Project, which is then divided into two *low-level containers*: the Back-End Requirements set and the Front-End Requirements set.



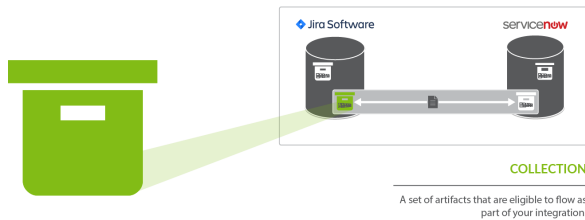
When we configure our Jama collection, we will define that collection at the *high-level container* level: this means that we can define the collection based on projects. Here, we have selected the Mobile App Project for use in our collection.



However, when routing artifacts, we will utilize *low-level containers* (sets) to determine which container Jama artifacts will flow to in our target repository. In the example below, the Back-End Requirements set in Jama will flow to the Back End Project in Jira, and the Front-End Requirements set in Jama will flow to the Front End Project in Jira. Both the Front End Requirements set and the Back End Requirements set are contained within the high level Mobile App Project, within Jama.

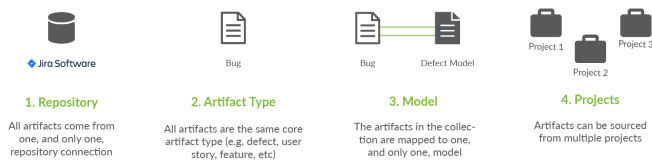


## Collection



A **collection** is the **set of artifacts that are eligible to flow as part of your integration**. They have the following characteristics:

1. All artifacts in the collection are the same core artifact type (e.g., defect, user story, feature, etc)
2. The artifacts in the collection are mapped to one model
3. Artifacts can be sourced from multiple projects (containers)



A concrete example of a collection would be a set of defects from an organization's *Jira* instance.

The artifacts in a collection can come from one or more projects from a given repository connection. Getting back to the example provided, if your *Jira* instance had 50 projects, you could include artifacts from any or all of those projects. Once projects are added to a collection, those artifacts are eligible for inclusion in an integration.

(💡 **Note:** The term **project** is used here generically — sometimes repositories have different names for **project**, or may not have more granular projects at all, but let's stick with this for simplicity's sake.)

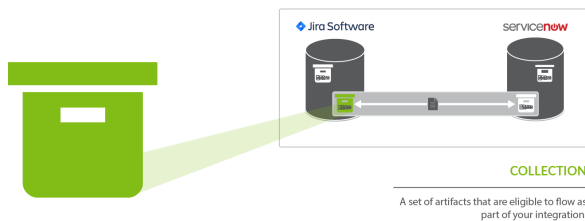
The artifacts in a collection share a set of fields that have repository-specific names and values. Part of creating a collection involves choosing a model on which to base the collection and then mapping these repository specific fields and values to those defined in the model. The concept of models will be discussed in the next section.

There are four types of collections in Tasktop:

- **Work Item Collections**, which typically include work items, such as requirements or defects, from typical repositories, such as *Jira* or *Micro Focus (HPE) Octane*. Work Item Collections can also be utilized to connect to a Database, such as *MySQL*, for use in an Enterprise Data Stream Integration
- **Container Collections**, which include certain container types from external repositories (such as Jama Components and Micro Focus/HPE ALM Folders)
- **Gateway Collections**, which contain information sent via an inbound webhook, from an external tool. Oftentimes, this information is generated based on an event, such as a failed test or a code review.
- **Outbound Only Collections**, which contains artifacts like code commits or changesets, where you may want to only flow out of your repository, but which would not receive updates into your repository.

You can learn how to create your collection(s) [here](#).

## Repository Collections (Work Item Collections and Container Collections)



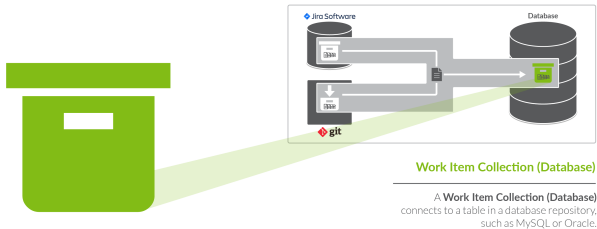
Repository Collections (meaning either a Work Item Collection or a Container Collection that connects to a repository) comprise artifacts from an ALM, PPM, or ITSM repository like *Atlassian Jira*, *ServiceNow*, *CA Clarity*, or *Zendesk*. When used in an integration, artifacts in a repository collection can be created, can be updated, and/or can trigger the creation of artifacts in another collection.

### What can Tasktop do to artifacts in a repository collection?

Action	Permissable
Create artifacts in collection	✓

Update artifacts in collection	✓
Detect additions or updates to artifacts in collection in order to create or update artifacts in another collection	✓

## Database Collections (a type of Work Item Collection)



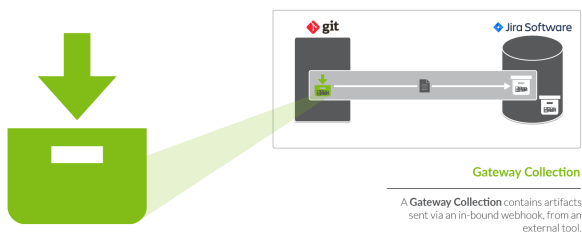
Databases collections (a type of Work Item Collection) connect to a table in a database repository, such as MySQL or Oracle. Artifacts in the source repository will flow data to the fields in that table.

When used in an integration, artifacts in a database collection can be created, but cannot be updated nor trigger the creation of artifacts in another collection.

## What can Tasktop do to artifacts in a database collection?

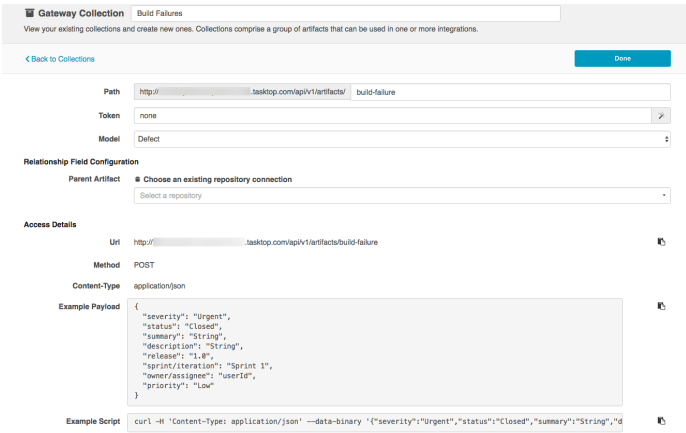
Action	Permissible
Create artifacts in collection	✓
Update artifacts in collection	✗
Detect additions or updates to artifacts in collection in order to create or update artifacts in another collection	✗

## Gateway Collection



Unlike repository collections and database collections, which rely on Tasktop actively making various API calls to communicate with a given repository, **artifacts in a Gateway collection are sent to Tasktop via our own REST API**. This means that you don't need to create a repository connection to create a gateway collection--as long as you can send Tasktop a simple REST call, those artifacts can then be used to achieve a specific goal within the context of an integration.

Gateway collections are particularly useful when the artifacts you want to integrate come from smaller, purpose-built systems for practitioners in various disciplines, such as Selenium for QA; when the artifacts you want to integrate come from systems that are largely event-driven, such as an application performance monitoring repositories; when artifacts come from home-grown tools your organization might have developed on their own; or when you'd like to pull information that is not considered a standard artifact from a repository supported by Tasktop, like capacity information from a PPM tool. When creating a gateway collection, you'll specify a path to generate a webservice to which you'll post information. You'll also choose the model to which you would like incoming artifacts from this collection to conform. You'll then be given an example payload and script that can be used to send artifacts to Tasktop:

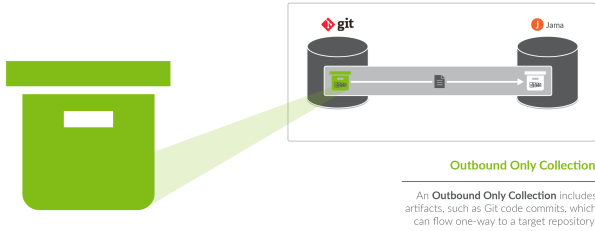


When used in an integration, artifacts in a gateway collection can trigger the creation or modification of artifacts in another collection.

### What can Tasktop do to artifacts in a gateway collection?

Action	Permissable
Create artifacts in collection	
Update artifacts in collection	
Detect additions or updates to artifacts in collection in order to create or update artifacts in another collection	

### Outbound Only Collections



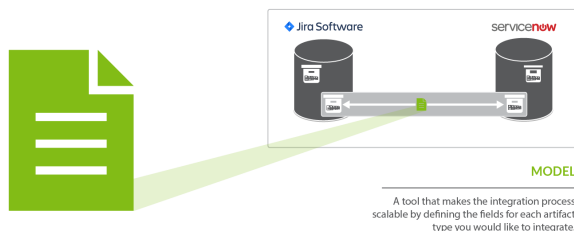
**Outbound Only Collections** contain artifacts like code commits or changesets from Source Code Management repositories like *Git*, which you may want to flow out of your repository, but which would not receive updates into your repository. When used in an integration, artifacts and information will be sent to a target repository. For example, you can create a Git commit in an ALM tool like *Atlassian Jira*. You can also update existing artifacts with information from the Git commit or changeset. While you can use this collection to flow artifacts or information out of a repository, the artifacts in this collection will not receive any updates.

**Note:** Outbound Only collections can connect to the Git repository only. You can learn more about configuring that repository in our [Connector Docs](#).

## What can Tasktop do to artifacts in an Outbound Only collection?

Action	Permissable
Create artifacts in collection	✗
Update artifacts in collection	✗
Detect additions or updates to artifacts in collection in order to create or update artifacts in another collection	✓

## Model



When integrating data from multiple collections, there are three factors that are critical to success:

1. The ability to normalize disparate definitions of artifacts between different collections
2. The ability to scale the integrations to support many collections with hundreds or even thousands of projects and artifacts.



3. Efficient flow of data – meaning, only flow information that is necessary between collections

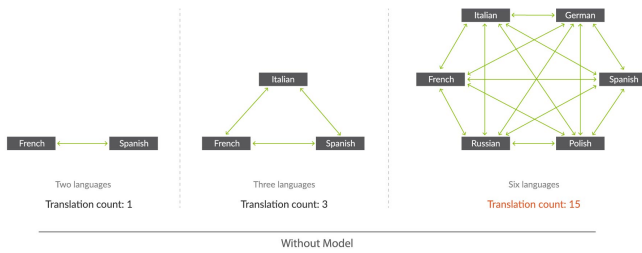
These three critical success factors are met with our usage of **models**. In very basic terms, **a model is simply a list of fields or attributes that define a certain artifact that you want to integrate**. For example, below is a very basic defect model:

Defect Model	
Field	Field Type
Description	String
Priority	Single Select: <ul style="list-style-type: none"><li>• High</li><li>• Medium</li><li>• Low</li></ul>
Status	Single Select: <ul style="list-style-type: none"><li>• New</li><li>• In Progress</li><li>• Complete</li></ul>

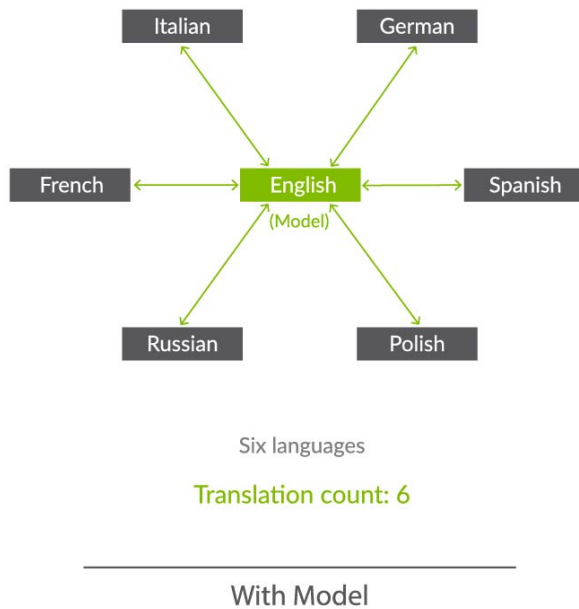
Let's talk about the first critical success factor – the ability to normalize disparate definitions of artifacts between different collections. Or, another way of thinking of it, the classic *“you say tomato, I say tomahto”* conundrum. In the diagram below it is apparent that the Jira bug is similar, but not the same, as the Jama defect. The solution to this problem is to be able to “map” each defect to a common definition of a defect and “normalize” the fields and field values. Then, when you are communicating about “defects”, everyone is speaking the same language via the *“model”* definition. Like this:

A good analogy to help understand why models are so important is the act of translating between people who speak different languages. If you have two people that speak two different languages, you need to translate only between those two points. If, however, you have three different languages, you have three points of disconnect in communication that need to be translated. But, as you add more and more languages, the number of disconnects blocking communication does not grow linearly – even if you have just 6 languages, you have 15 points of disconnect to translate between! And if you have 10 languages you will have 45! As you can see, resolving these point-to-point disconnects individually quickly becomes unsustainable given the sheer number of them that can arise. **It is in this way that models save the day, acting as a “universal translator,” overcoming all of the communication disconnects that are present by translating between all of the points at once.** Now that we have the ability to solve the *“you say tomato, I say tomahto”* problem, the second critical success factor comes into play, which is the desire to *scale your integration landscape* to support many collections with hundreds or even thousands of projects and artifacts.

## Integrating Without Models



## Integrating With Models



Now that we've solved the first two critical success factors, there is one more that might not seem as obvious but is actually quite important to your overall success. When flowing large volumes of data, you need *efficient flow of data*, not the 'drink from the firehose' approach where all fields of all artifacts are flowing everywhere. There is no business value in that and, worse, you will end up with significant performance issues. Instead, by using *models*, you can limit, or target, the exact data that you need to flow between collections – nothing more, and nothing less, than what is necessary.

In summary, models solve the critical three success factors for large-scale integration landscapes – giving users the ultimate in flexibility, scalability, and consistency at the same time.

You can learn how to create a model [here](#).

## Flow Specification & Templates

Now that we have introduced the concepts of **artifacts**, **collections**, and **models**, we can come back to the concept of an **integration**. As discussed earlier, the basic concept of an integration is the **flow of information between two or more collections**.

The last two concepts to introduce relate to integrations as a whole. First, the **flow specification**. This is probably the trickiest aspect of an integration, which is why we also have introduced another concept, called **templates**, to help.

Defining how you'd like data to flow between collections requires a lot of nuance and forethought. For instance, would you like to create new artifacts, or modify existing artifacts? Would you like artifacts and fields to flow in both directions or just one direction? What types of collections (and how many of them) would you like to integrate?

**Picking a template jump-starts your integration, bundling many of the flow specification elements to facilitate quicker configuration.**

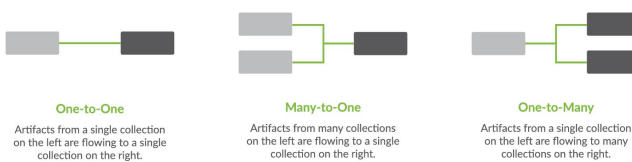
You can learn how to configure your integration using a template [here](#).

## Integration Style

Each template is based on an underlying style that defines whether you want to create new artifacts in collections or modify already existing artifacts in collections.

## Canvas Layout

Each template follows a certain canvas layout, determining the quantity and types of collections that can be added to the canvas. The canvas will either follow a many-to-one, one-to-many, or one-to-one layout.



By picking a given template, you are, in essence, also picking the style of integration and canvas layout, which in turn influences other configuration options such as the artifact flow directionality, field flow directionality, and routing directionality, making the act of integrating your collections quick and painless.

## Artifact Relationship Management

**Artifact Relationship Management (ARM)** refers to the ability to maintain relationships between artifacts when they flow from one collection to another. By utilizing the Relationship Specification Screen when configuring your collection, you can ensure that relationships are preserved between your artifacts. You'll learn more about how to configure ARM in the [Quick Start Guide](#).

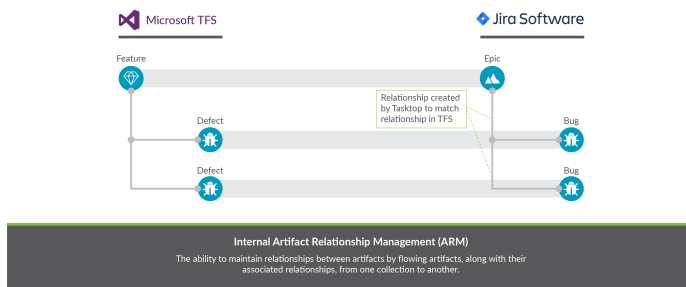
# Internal ARM

When using Tasktop, it is important to understand the distinction between Internal ARM and External ARM.

Internal ARM refers to the ability to flow multiple artifacts between two (or more) repositories, and to maintain relationships between them.

In the example below, you can see an example of an Integration from Microsoft TFS to Jira which utilizes ARM to do the following:

- Flow Microsoft TFS Features to Jira Epics
- Flow Microsoft TFS Defects to Jira Bugs
- Utilizes Artifact Relationship Management (ARM) to preserve the relationships between the artifacts internally within each repository

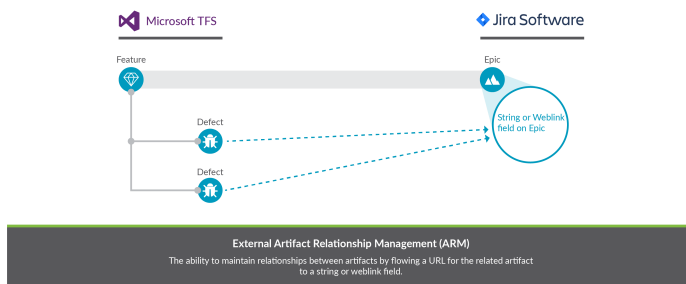


# External ARM

External ARM is a more light-weight approach, compared to internal ARM. Rather than flowing the related artifacts themselves to the target repository, you can flow a link to those artifacts to a string or weblink field.

For example, you could:

- Flow Microsoft TFS Features to Jira Epics
  - The Microsoft TFS Features are 'affected by' defects within TFS
- Instead of flowing the TFS Defects to Jira, we can flow a link to those TFS defects to a string or web link field on the Jira Epic



## Key Concepts Video

You can learn more about these concepts in the short video below:

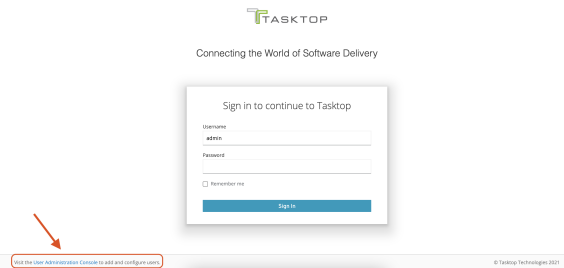
# User Management

## Getting Started

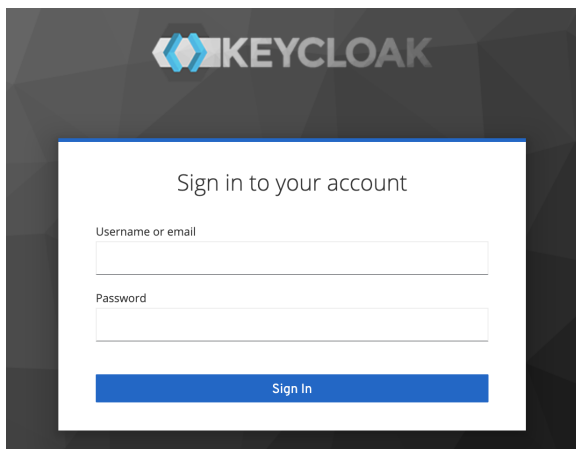
*Tasktop Cloud users will access user administration directly via the Tasktop UI and not in the external User Administration Console.*

Once [installation](#) is complete, you can begin using Tasktop Hub by opening <http://localhost:8080/> or <https://localhost:8443/> in any of our [supported browsers](#).

Before logging in to Tasktop Hub, you must log in to the **User Administration Console** to create your admin user(s). This can be accessed via the **User Administration Console** link at the bottom of the Tasktop login screen.



After clicking the link, the Keycloak login screen will appear.



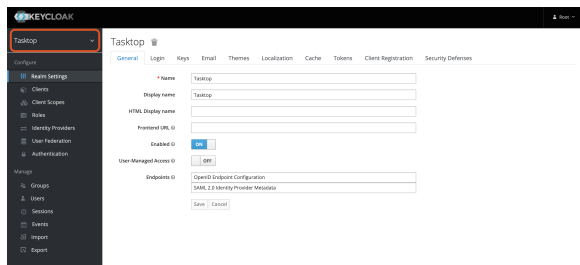
The Tasktop User Administration Console comes pre-configured with a root user and password. You can use the following credentials to log in to Keycloak:

- **Username:** root
- **Password:** Tasktop123

**⚠ Note:** There is only **one** initial root user. If the credentials for this user are lost, access to the [advanced User Management](#) features will also be lost. All functionality of Tasktop Hub will continue uninterrupted. You can learn how to create additional root users and manage existing root users [here](#).

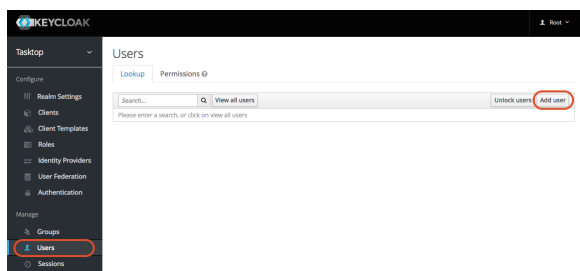
After logging in, you will be prompted to change your root password and you will need to make at least **one** new Tasktop Admin user for Tasktop. After this first user is created, you can create additional users directly from the Tasktop interface.

To create a Tasktop Admin, ensure the **Tasktop** realm is selected here:

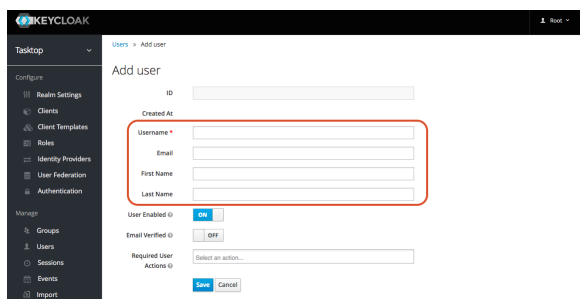


**Note:** Do not rename the realm (**Tasktop**), as this will result in errors upon Tasktop log in. If you must rename the realm, please also edit `{tasktop workspace}/webapps/ROOT/WEB-INF/keycloak.json`, update the 'realm' parameter, and then restart Tasktop.

Select the **User** section in the left column and click **Add user**.



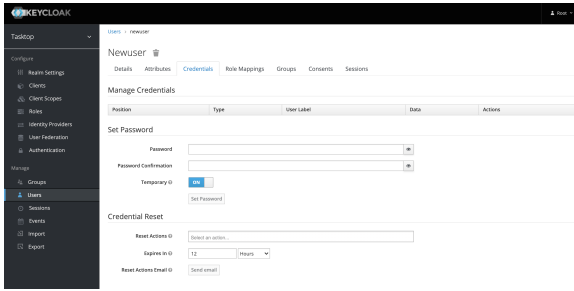
On this screen, enter the username, email, first name, and last name for your new user — the rest of the fields are **not** required. Once you've entered the required fields, click **Save**.



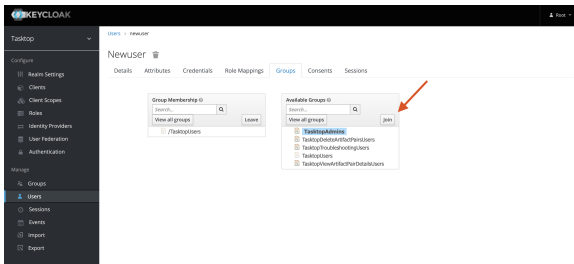
After you have saved the user, select the **Credentials** tab and provide a temporary password for the new user.

**Tip:** Please ensure the Temporary toggle is set to **On**. This will allow the user to set a new password upon their first log in.

Once you have entered the temporary password, click **Set Password**.



Next, select the **Groups** tab to assign the user as a Tasktop Admin. Highlight **TasktopAdmins** and click **Join**. By becoming a Tasktop Admin, the user can add new users directly from the Tasktop Hub interface.



**Tip:** You can ignore the Attributes, Role Mappings, Consents and Sessions tabs.

That's it! Your Tasktop Admin user has been added.

Now, you can sign out of the User Administration console, navigate to <http://<server>:8080>, and log in with your newly created user account.

## Types of Users

There are several types of users in Tasktop Hub:

- **Users:** This user has all permissions needed to create, modify, and run integrations.
- **Admins:** This user has the same permissions as a **User** and also includes the following permissions:
  - Create new users
  - Update users' passwords
  - Change users' group membership (from user to admin or vice-versa)
- **Troubleshooting Users:** This user can review Tasktop errors, logs, usage reports, and configurations, but cannot alter Tasktop integration configurations or user management.
  - **Note:** Troubleshooting Users were added in Tasktop version 19.4, and may require additional steps if you'd like to update specific settings. For more information on configuring the Troubleshooting User role, please see the section [below](#).
- **View Artifact Pair Details Users:** This user can view artifact pair details.
- **Delete Artifact Pair Users:** This user can delete artifact pairs.

**Note:** Any users upgrading from versions prior to 21.1 will need to follow the steps outlined [here](#) for the Artifact Pair user roles to appear. All users installing Hub after 21.1 will have the Artifact Pair user roles by default and will not need to follow any additional steps.



## Best Practices

We recommend configuring **at least two admin users** — that way if one admin forgets their password, the other admin can log in and reset the other admin user's password.

We also recommend changing the default password of the **Advanced User Administration** console. See the [Getting Started](#) section above for information on how to reset passwords.

## User Permissions


Capability	Admin	User	Troubleshooting User	View Artifact Pair User	Delete Artifact Pair User
Create New User	✓	✗	✗	✗	✗
Reset Any User's Password	✓	✗	✗	✗	✗
View and Modify Any User's Group Membership	✓	✗	✗	✗	✗
Reset Own Password, Name, or E-mail	✓	✓	✓	✗	✗
Create and Modify Repository Connections	✓	✓	✗	✗	✗
Create and Modify Models	✓	✓	✗	✗	✗
Create and Modify Collections	✓	✓	✗	✗	✗
Create, Modify, and Run Integrations	✓	✓	✗	✗	✗
Download Troubleshooting Reports (logs, usage reports, etc)	✓	✓	✓	✗	✗
Change Logging Frequency	✓	✓	✓	✗	✗
Review Errors & Configurations	✓	✓	✓	✗	✗
Retry, Prioritize, and Recreate Errors	✓	✓	✗	✗	✗
View artifact pair details	✗	✗	✗	✓	✗
Delete artifact pairs	✗	✗	✗	✗	✓
Access to <code>/api/v1/integrations/delete-integration-data</code> public API	✓	✗	✗	✗	✗

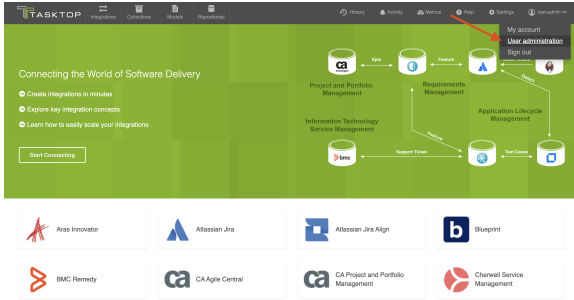
Access to `/api/v1/integrations/delete-all-integration-data` public API



## Creating Additional Users

To create a user, select **User Administration**.

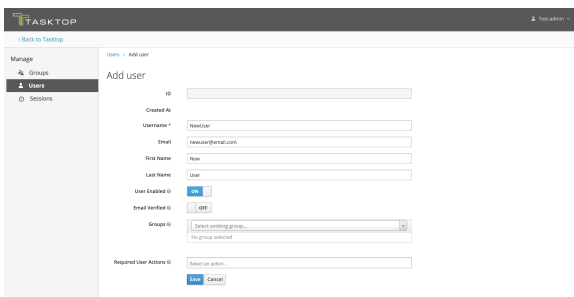
 **Note:** You must have **admin** capabilities to create an additional user.




From the **User Administration** screen, select **Add user**.



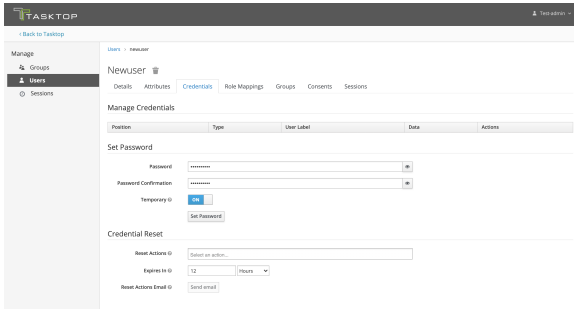
On the **Add User** screen, enter the username, email, first name, and last name for your new user — the rest of the fields are **not** required. Once you've entered the required fields, click **Save**.



After you have saved the user, select the **Credentials** tab and provide a temporary password for the new user.

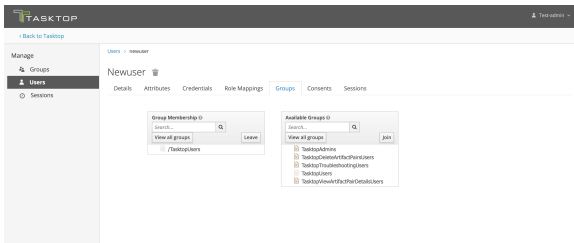
 **Note:** Please ensure the Temporary toggle is set to **On**. This will allow the user to set a new password upon their first log in.

Once you have entered the temporary password, click **Set Password**.



Next, click the **Groups** tab and add the user to a group — based on the permissions you'd like the user to have.

**Note:** If the new user is not added to a group, they will not be able to successfully access Tasktop Hub.



**Tip:** You can ignore the Attributes, Role Mappings, Consents, and Sessions tabs.

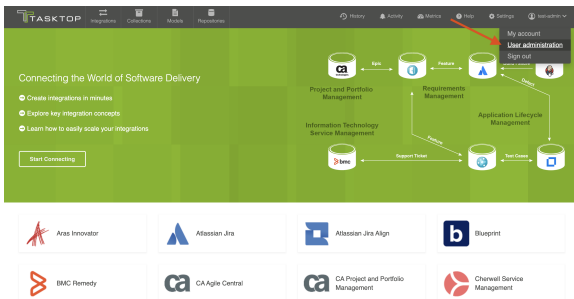
That's it! Your user has been added and can log in with their temporary password.

**Note:** Tasktop will not send the new user an email notification. The **admin** must notify the user of the new account and password.

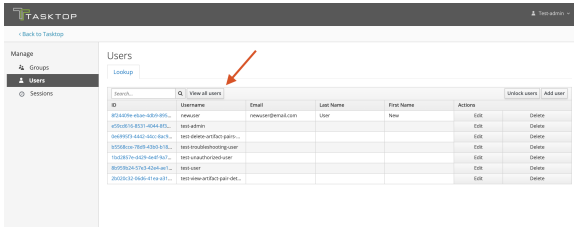
## Resetting a User's Password

To reset a user's password, select **User Administration** from the upper right corner of the application.

**Note:** You must have **admin** capabilities to reset a user's password.



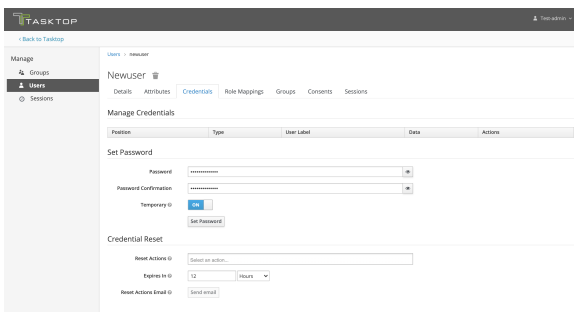
Click **View all users**. Next, click on the **ID** for the user whose password you'd like to reset.



Then, click the **Credentials** tab and provide a temporary password for the user.

**Note:** Please ensure the Temporary toggle is set to **On**. This will allow the user to set a new password upon their first log in.

Once you have entered the temporary password, click **Set Password**.



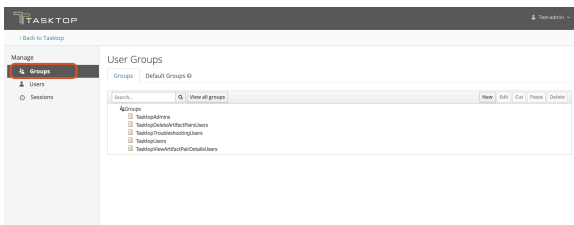
**Note:** Tasktop will not send the user an email notification. The **admin** must notify the user of the new temporary password. The user will be prompted to set a new password upon their next log in.

## Managing Groups

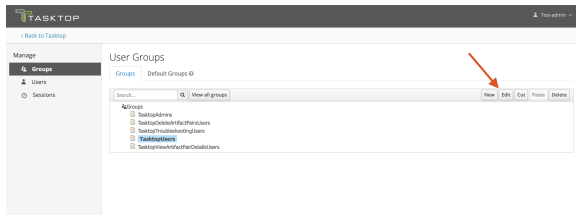
### Viewing Members of a Group

To view the members of a group, click **Groups** on the left side of the **User Management** screen.

**Note:** You must have **admin** capabilities to view members of a group.

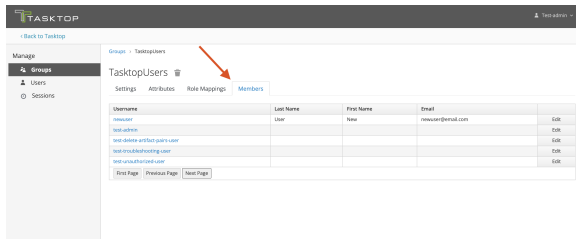


Next, select the group you'd like to review, and click **Edit**.



To view the group's current members, click the **Members** tab.

**Tip:** A user can be a member of multiple groups.



## Adding or Removing Users From a Group

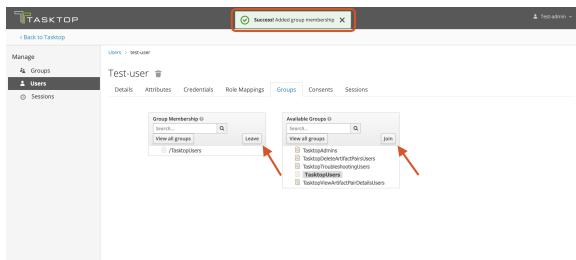
Select **Users** from the left pane of the **User Administration** screen. Click **View all Users** and select the **ID** of the user you'd like to modify.

**Note:** You must have **admin** capabilities to modify a user's group membership.

Click the **Groups** tab and select the group whose membership you'd like to modify. Then, use the **leave** and **join** buttons to modify their group membership.

There is no saving necessary here. Once you click **leave** and/or **join**, you will see a notification at the top of the screen informing you that your change has been made.

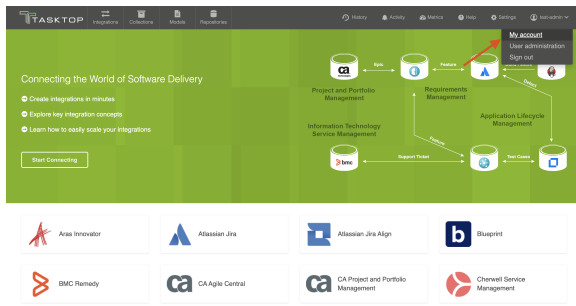
**Note:** A user must be a member of at least **one** group in order to be able to log in to Tasktop successfully.



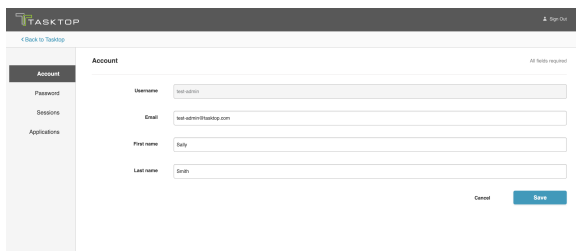
## Modifying Your Own User Information

To change your own password or other user information, click your username at the upper right corner of the screen, and select **My Account**.

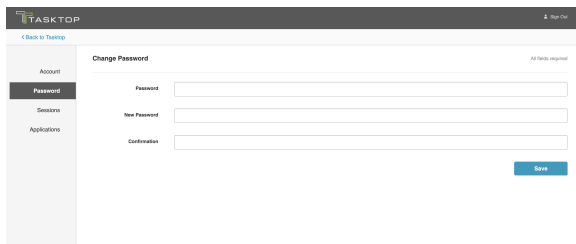
 **Tip:** Both **users** and **admins** can modify their own account information.



This will bring you to the **Account Info** screen, where you can update your name or email address.



Click **Password** on the left sidebar to change your password.

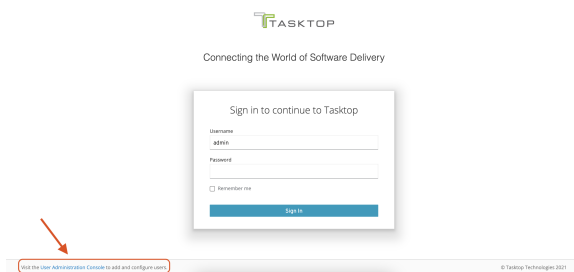


 **Tip:** The **Sessions** and **Applications** sections can be ignored.

## Advanced User Management

Tasktop Hub has advanced user management capabilities that are not accessible via the Tasktop Hub interface.

To access advanced user management capabilities, click the **User Administration Console link** at the bottom of the Tasktop Hub login screen.



You can log in using the credentials you set when you [first installed and began using Tasktop](#).

**⚠ WARNING:** There is only one initial root user. If the credentials for this user are lost, access to the advanced User Management features will be lost. All functionality of Tasktop, however, will continue uninterrupted.

Some of the advanced features include:

- User Federation Configuration for:
  - LDAP
  - Kerberos
- Identity Provider login for:
  - SAML v2.0
  - OpenID Connect v1.0
- Enforcing custom password policies such as:
  - Set password expiration
  - Require special characters
  - Setting minimum password length

**⚠ Note:** While Tasktop officially supports LDAP, other advanced features (including but not limited to Kerberos Federation and IDP) are not supported or tested by Tasktop.

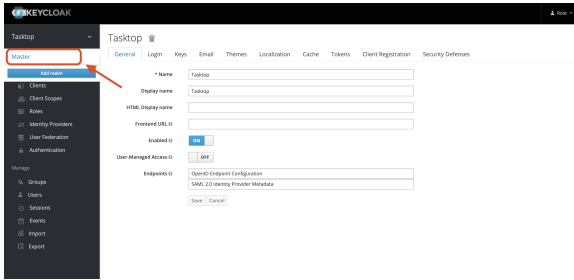
To learn more about these advanced features, click [here](#).

**⚠ WARNING:** Do not make changes or updates to the Roles or Groups section. Altering these settings may prevent your Tasktop Hub users from accessing the tool.

## Creating and Managing Root Users

A **root user** refers to a user who can log in to the **User Administration Console**. Tasktop comes with one root user, but if you'd like to create additional root users or to manage existing users, you can do so from the **User Administration Console**.

Once logged in, click the arrow next to **Tasktop** (in the upper left panel) and select **Master**.



Next, click **Users** in the left panel.

From here, you can follow the [same instructions used to create Tasktop users](#) to create and manage root users (ignoring the **Groups** section).

## Configuring the Troubleshooting User



This section is only applicable when upgrading from versions earlier than Tasktop Hub 19.4.

### Creating the Troubleshooting User Role using a Script

To configure the troubleshooting user role, we provide a script that will create the **TasktopTroubleshootingUser** role in your Keycloak instance, and replace the default **TasktopUsers** group with the **TasktopTroubleshootingUsers** group.

**Note:** This script can only be used if you have provided a valid SSL certificate as described in the [SSL Certificate Installation](#) section. If you have not provided such a certificate, skip to the **Creating the troubleshooting user role via the Keycloak admin console** section below.

#### Windows

Run the `add-troubleshooting-user.bat` script in `C:\Program Files\Tasktop\utility-scripts`, providing the relevant information when prompted.

#### Linux

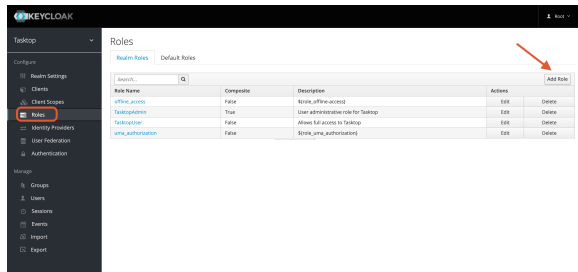
Run the `add-troubleshooting-user.sh` script in `<installation location>/Tasktop/utility-scripts`, providing the relevant information when prompted.

### Creating the Troubleshooting User Role via the Keycloak Admin Console

If you have not provided a valid SSL certificate, you can create a troubleshooting user via the **User Administration Console**. This console can be accessed by following the instructions in the [Getting Started](#) section.

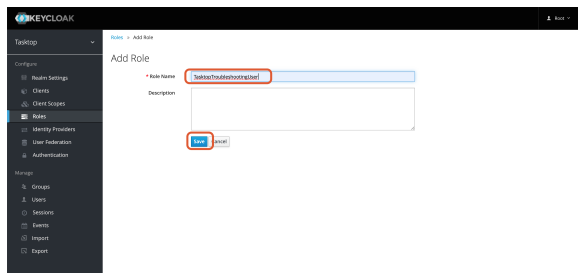
After logging in, navigate to the **Roles** section in the left column and click **Add Role**.





On this screen, enter **TasktopTroubleshootingUser** in the Role Name field. Then, click **Save**.

**Tip:** The Role Name is case-sensitive and must match exactly.

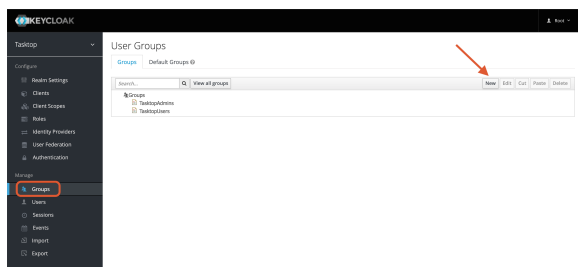


That's it! The troubleshooting user role has been created. Next, you'll need to add the troubleshooting user to a group.

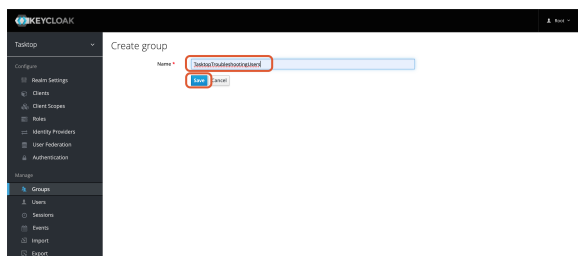
## Adding Troubleshooting Users to a Group

We recommend that you create a group for troubleshooting users and set it as the default group.

To do this, navigate to the **Groups** section in the left column and click **New**.

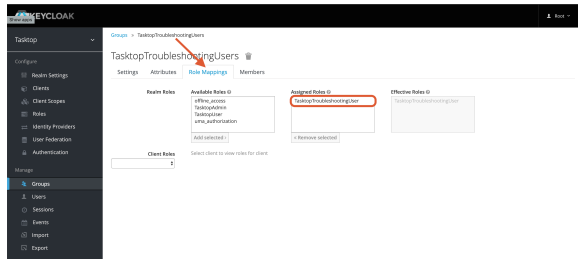


On the Create Group screen, enter **TasktopTroubleshootingUsers** in the Name field. Then, click **Save**.



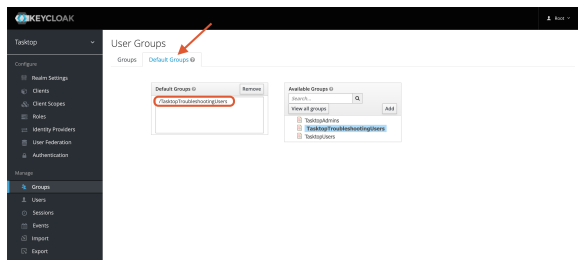
After saving the group, the new group screen will appear.

Next, select the **Role Mappings** tab and add **TasktopTroubleshootingUser** to Assigned Roles.



After you have added the user to Assigned Roles, navigate back to the **User Groups** screen and select the **Default Groups** tab.

Next, remove any groups under **Default Groups** and add the **TasktopTroubleshootingUsers** group.



This section is only applicable to Tasktop Hub version 19.4 - 21.1.

Upon installation, new users will default to having the **TasktopUser** role. If you'd like to set the default to **TasktopTroubleshootingUser**, please follow either set of instructions below.

## Setting the Default Troubleshooting User Group Using a Script

To configure the troubleshooting user role, we provide a script that will create the **TasktopTroubleshootingUser** role in your Keycloak instance, and replace the default **TasktopUsers** group with the **TasktopTroubleshootingUsers** group.

**Note:** This script can only be used if you have provided a valid SSL certificate as described in the [SSL Certificate Installation](#) section. If you have not provided such a certificate, skip to the **Creating the troubleshooting user role via the Keycloak admin console** section below.

### Windows

Run the `add-troubleshooting-user.bat` script in `C:\Program Files\Tasktop\utility-scripts`, providing the relevant information when prompted.

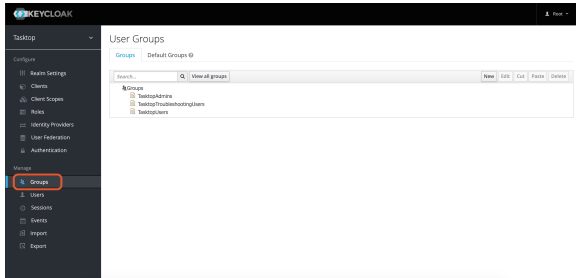
### Linux

Run the `add-troubleshooting-user.sh` script in `<installation location>/Tasktop/utility-scripts`, providing the relevant information when prompted.

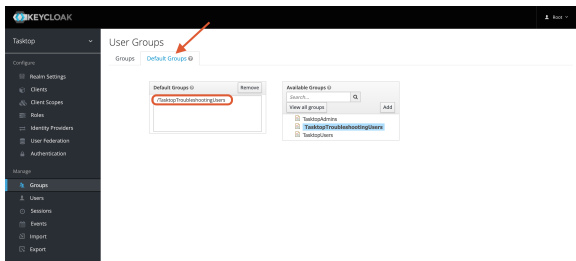
## Setting the Default Troubleshooting User Group via the Keycloak Admin Console

If you have not provided a valid SSL certificate, you can set the troubleshooting user group as the default via the **User Administration Console**. The console can be accessed by following the instructions in the [Getting Started](#) section.

After logging in, navigate to the **Groups** section in the left column.



Select the **Default Groups** tab. Remove any groups under **Default Groups** and add **TasktopTroubleshootingUsers**.



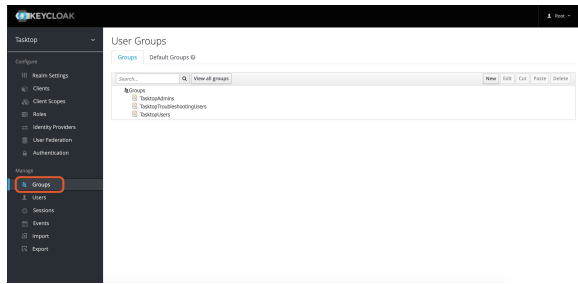
This section is only applicable to Tasktop Hub version 21.2 and later.

Upon installation, new users will default to having the **TasktopUser** role. If you'd like to set the default to **TasktopTroubleshootingUser**, please follow the instructions below.

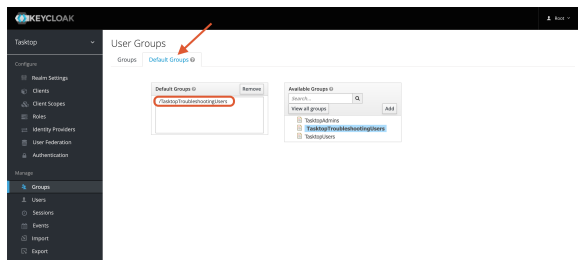
## Setting the Default Troubleshooting User Group via the Keycloak Admin Console

If you have not provided a valid SSL certificate, you can set the troubleshooting user group as the default via the **User Administration Console**. The console can be accessed by following the instructions in the [Getting Started](#) section.

After logging in, navigate to the **Groups** section in the left column.



Select the **Default Groups** tab. Remove any groups under **Default Groups** and add **TasktopTroubleshootingUsers**.



## Configuring LDAP User Management

### Required Directory Information

Before configuring LDAP, please check you have the following required pieces of information available for your specific Active Directory (AD) domain.

- The **fully qualified domain name (FQDN)** for the AD service,
  - *example: 'demo.tasktop.com'*
- An AD **user** account and credentials; The user will need read / view access to Users, Groups and Organizational Units (OU). We suggest a specific restricted account be setup in AD for this purpose.
  - *example: 'service\_tasktop'*
- An AD user **group**; The group(s) will be used to store specific users, who will have access to Tasktop.
  - *example: 'Tasktop Users'*
- A tool such as **ADSIEdit**, which is able to give your the specific information about the structure of your AD domain setup.
  - **ADSIEdit** is part of Microsoft Windows Remote Server Administration Toolset (RSAT). This can be downloaded from [Microsoft RSAT page](#), or enabled on a server by adding the RSAT feature.
  - Alternatively, ask your Domain Administrators for all of the following information:
    - CN/DN for Tasktop User (mentioned above)
    - CN/DN for the Tasktop User Group (mentioned above)
    - User, mail; username and name attributes (the specific name for each attribute)
    - OU root for all users
    - LDAP FQDN server URL

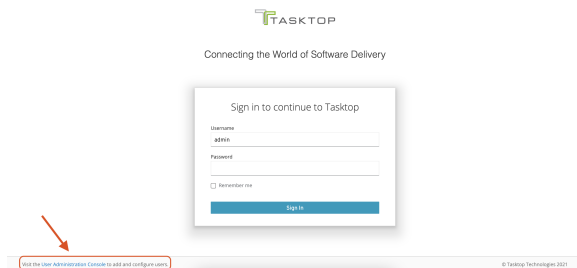
## Importing SSL Certificates

If you would like to connect to an LDAP server, you will need to import the SSL certificate into the keystore of your Tasktop product and restart it. To import the certificate to the keystore, see the following:

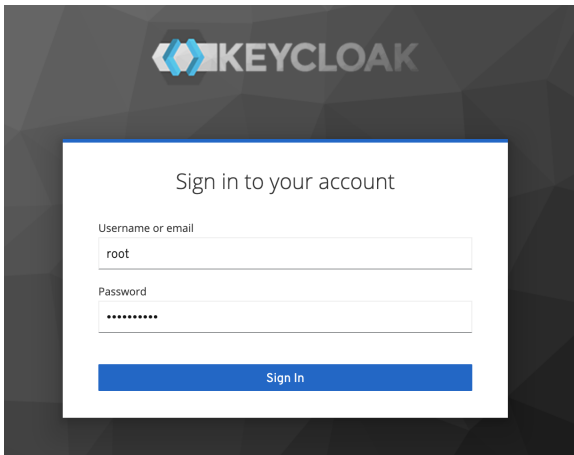
- Shut down your Tasktop instance (including Keycloak)
- Obtain the certificate and certificate chain for your LDAP server. You may be able to do this using a command like the following on Linux
  - `echo -n | openssl s_client -connect <ldap-server>:636 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ldapserver.pem`
- In command prompt enter the following:
  - `<path_to_jre>/bin/keytool -import -trustcacerts -keystore <path_to_keystore> -storepass <password> -alias ldap -file ldapserver.pem`
    - `<path_to_jre>` refers to the jre folder in the Tasktop install location.
    - `<path-to-keystore>` refers to the path to the truststore referenced [here](#).
    - `<password>` is the password of your keystore or change it if you are using the default
    - the keytool command should be run for each certificate exported. Each will need to have a unique alias.
  - the default password is: changeit
- Start your Tasktop product.
- Try again to connect to LDAP Server.

## Accessing Keycloak Configuration Tool

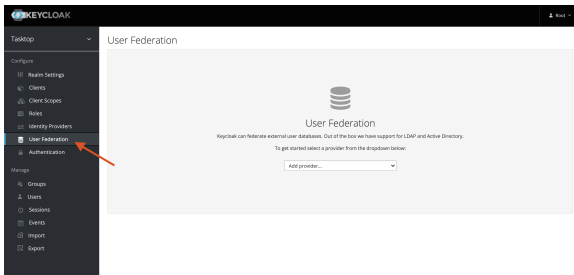
1. To access advanced user management capabilities, click the **User Administration Console** link at the bottom of the Tasktop Hub login screen.



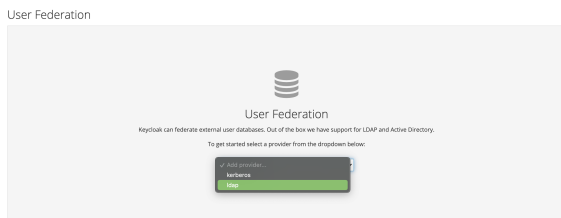
2. Log in using the default credentials listed in the [Getting Started](#) section above.



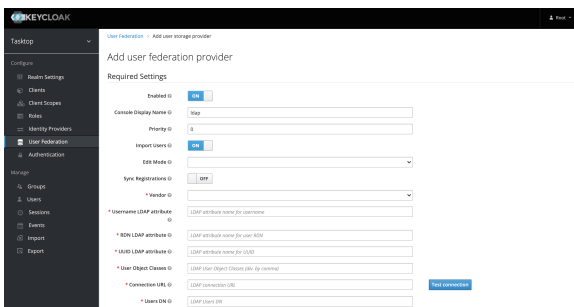
3. Select the **User Federation** link from the left side panel.



4. Choose the **ldap** option from the dropdown for **Add provider...**



5. The **LDAP configuration** screen should now be displayed.



## Configuring LDAP for Active Directory

This section will guide you through creating a connection to an LDAP authentication server.

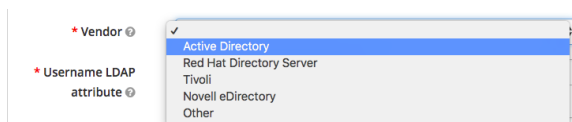
**Note:** Images provided are only a sample of settings — please ensure that you enter information specific for your environment.

## Required Settings

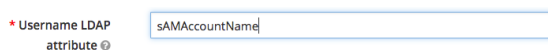
**Tip:** Follow the steps [above](#) to access the **LDAP configuration** page.

See the required settings below:

- **Console Display name:** This is the name you'd like to give your connection.
- **Priority:** If you have more than one User Federation configured, this setting specifies in which order to search each user federation service, **0** is first.
- **Edit Mode:**
  - **READ\_ONLY:** This setting reads the attributes from Active Directory (AD). It will not attempt to modify the AD service or store any local changes to user information.
  - **WRITABLE:** This setting may enable some changes to be written back to AD. The user account communication with AD will need access to modify the specific objects attribute.
  - **UNSYNCED:** This setting reads the attributes from AD and synchronizes them to a local store in the internal Keycloak database. **Users** and **Administrators** can make changes to the user objects, but those changes will only be stored for the local Tasktop instance. This will not write back to Active Directory.
  - **Tip:** The recommended mode is **READ\_ONLY**.
- **Sync registrations:** If a new user is created in Tasktop, this will allow that user to also be created in AD if you have **WRITABLE** selected and access to create user objects in the AD domain. The default setting is **OFF**.
- **Vendor:** Specifies which vendor software to use for this LDAP configuration. If you are using something other than Active Directory, the attributes and locations may be different. This will also pre-fill some default values.



- **Username LDAP attribute:** This should be the default username attribute as specified in your domain. The default for Microsoft AD is **sAMAccountName**.



- **RDN LDAP attribute:** The Relative Distinguished Name LDAP attribute is a list of attributes which will be searched when a user attempts to authenticate to Tasktop. The attributes listed here should be unique within an OU level or unique within a domain. The following options are a good base to use:
  - **cn** (canonical name): the full name (e.g., *John Doe*)
  - **sAMAccountName:** the username (e.g., *john.doe*)

- **mail:** the email address (e.g., *john.doe@demo.tasktop.com*)

\* RDN LDAP attribute ⓘ

- **UUID LDAP attribute:** The User Unique Identification attribute is a complicated long string of characters which uniquely identify a single object within AD. For unix based LDAP this is often **uid**. The default for Microsoft AD is **objectGUID**.


\* UUID LDAP attribute ⓘ

- **User Object Classes:** These are the 'types' of objects which can be used to authentication against. You can specify more if your organization has other specific identifiers such as 'staff' or 'contractor'. The default for Microsoft AD is: **person, organizationalPerson, user**.

\* User Object Classes ⓘ


- **Connection URL:** This is the specific string which should be the FQDN of your LDAP service. It's default format for AD will be 'ldap://demo.tasktop.com'. If you have SSL configured then you can also use ldaps://demo.tasktop.com (SSL is not enabled by default in Microsoft AD).

\* Connection URL ⓘ

 **Tip:** At this point, we recommend selecting the **Test connection** button to check that Tasktop is able to communicate with your LDAP server. You should see a green message at the top of your screen indicating a successful connection to your LDAP server.

- **Users DN:** This is the Distinguished Name for the location where you can find your users. You can find the Users DN (and any other Distinguished Names) via the **ADSIEdit** tool in Windows. Once the tool is open, you will need to connect to the AD domain for your organization. Once connected, the domain will be presented in a tree-view on the left, where you can drill down to the specific branches until you find the specific OU or User object you want details for. We recommend using this utility as it will allow you to copy/paste the specific DN information directly (any typing mistakes will result in error when testing).

The format for this string will be a number of **OU=** followed by a number of **DC=** separated by a comma.

 **Tip:** Spaces are allowed in this string if they exist in your structure.

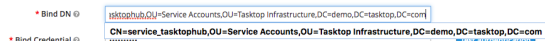
\* Users DN ⓘ

- **Authentication Type:** If using Microsoft Active Directory, you will be required to authenticate. Some non-Microsoft systems do not require authentication— if this is the case, select **none**.
- **Bind DN:** This is the Distinguished Name for the user account which you will use to authenticate against your LDAP service to allow Tasktop to authenticate users. The Bind DN user account can be anywhere within the AD domain, however, we suggest that you have a dedicated account



specifically for Tasktop. The format for this string will be a singular **CN=** for the Canonical Name of the user account, followed by possible **OU=** which is followed by the **DC=** items all separated by a comma.

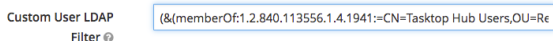
**Tip:** Spaces are allowed in the string if they exist in your structure.



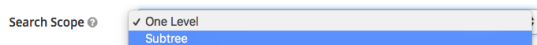
- **Bind Credential:** This is the password for the user account configured in the Bind DN.

**Tip:** Once you have entered the password, click **Test authentication** to confirm that Tasktop is successful in authenticating itself against your Active Directory domain. You should see a green message at the top of your page as an indication of a successful authentication.

- **LDAP Filter:** This is where you will configure a filter to specify which user accounts will have access to authenticate in Tasktop. If you leave this blank, all users within your **Users DN** OU in the AD environment will have access. The structure of the string is as follows:
  - () : braces to start and finish
  - Either
    - &() : for performing an 'AND' operation (i.e., all items must match)
    - |() : for performing an 'OR' operation (i.e., where any items can match)
  - Specific attribute related condition (e.g., matching objects in a group)
  - Users in a specific group can use **memberOf=**
    - *memberOf=CN=Tasktop Hub Users,OU=Resource Groups,OU=Groups,OU=Tasktop,DC=demo,DC=tasktop,DC=com*
  - Users and (nested) Groups in a specific group require **memberOf:1.2.840.113556.1.4.1941: =**
    - *memberOf:1.2.840.113556.1.4.1941:=CN=Tasktop Hub Users,OU=Resource Groups,OU=Groups,OU=Tasktop,DC=demo,DC=tasktop,DC=com*
  - You can also specify that a particulate attribute is equal to some value (e.g., *objectCategory= Person*)



- **Search Scope:** The Configuration of this depends on whether you have all of your AD users in a single OU, or if you'd like to search through the OU hierarchy structure. If searching, the Users DN field configured above will need to be the root or lowest-level OU.
  - If all users are in a single OU, set this to **One Level**.
  - If users are hierarchically organized in OUs, set this to **Subtree**.



- **Use Trusted SPI:** This is used if your environment uses SSL and a client certificate is required. This is not a default AD configuration.
- **Connection Pooling:** This will allow connections to your AD server to remain open if set to **ON**, (for specific timeframe) rather than creating a new connection each time a user authenticates.


- **Pagination:** This allows you to page (or cache) information for active connections from your AD servers.
- **Mappers:** Go to the **Mappers** tab at the top of the LDAP user federation you just created. Click **Username LDAP attribute**. Ensure that **LDAP Attribute** is the same as what you entered in **Username LDAP attribute** [here](#).

## Kerberos

 **Note:** Tasktop does **not** include instructions for Kerberos setup.


## Sync Settings

- **Batch Size:** Indicates how many accounts will process at once
- **Periodic Full Sync:** Allows for a sync of all users to occur between Tasktop and Active Directory. If you have a large number of users constantly authenticating into Tasktop, it may be useful to enable this. Default is set to **OFF**.
- **Periodic Changed Users Sync:** Allows for newly created or updated users to be synced from Active Directory to Tasktop. If you have the Periodic Full Sync enabled, you should also enable this. Default is set to **OFF**.

 **Tip:** Save your configuration by clicking **Save** at the bottom of the page. A green message at the top will indicate that your save was successful.

## Additional LDAP Information

### Testing

 **Note:** The configuration utility for LDAP requires its own internal authentication. As such, when you test account access it is recommended that you use a separate browser or select a **private** or **incognito** browser. If you are already logged in to Tasktop, you will first need to log out before testing.

1. Direct your browser to the default web address of your Tasktop server, such as **https://demo.tasktop.com/**
2. Enter credentials which should be allowed access to authenticate from the LDAP connection you have just setup
3. Retry with a set of credentials which should **not** have access to Tasktop. If you are able to log in, check the **filter** settings again.

### Default User Access

By default, all LDAP users will be granted **user** level access to Tasktop. If you have configured the troubleshooting user functionality (by running the script or performing manual configuration through the admin console), LDAP users will by default be granted **troubleshooting user** level access instead. If desired, you can set all new accounts, including LDAP user accounts, to default into a specific group. You can also assign different **members** to either of the **TasktopUsers** or **TasktopAdmins** groups.

To change the default group, use the following instructions:

1. Select **Groups** (under the **Manage** section) of the right-side bar menu.
2. Select the **Default Groups** tab.
3. Add or Remove the **TasktopUsers** and/or the **TasktopAdmins** groups to the **Default Groups** list.

## User Management and Security Constraints

Tasktop's User Management uses Security Constraints as described in the Java Servlet Specification to limit access to authenticated users. Adding additional Security Constraints to the Apache Tomcat configuration can interfere with Security Constraints provided by Tasktop and enable unauthenticated users to access Tasktop.

## DNS Settings


The server Tasktop is installed on must be able to resolve the hostname clients will use to access it. This can be accomplished through the DNS configuration. A less preferred option is to configure using the server's hosts file.

The hostname clients use to access Tasktop must be a valid hostname according to RFC 952. This means it may only contain letters, digits, hyphens, and periods, and may not contain underscores.

## Alternative User Management

By default, Tasktop comes with a user management solution. In the rare scenario where your organization decides not to use Tasktop's provided user management solution and you still need to ensure that only authorized users are able to access your Tasktop instance, you can set up Basic Authentication for the Tomcat web server.

Additional information on configuring Tomcat authentication can be found [here](#).

 **Note:** Using this style of user management will mean that all of your users will have the exact same permissions within Tasktop. There will be no separate roles or permissions within the application.

# Quick Start Guide

## Overview

[Overview](#) | [Connect to your Repository](#) | [Create or Reuse a Model](#) | [Create your Collections](#) | [Configure your Integration](#) | [Running your Integration](#)

This quick-start guide walks you through setting up a basic integration, including essential steps like connecting to your repository, constructing your model, and creating your collections.

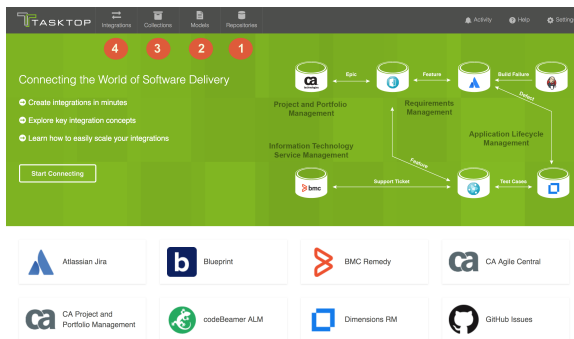
Whether you're just getting started with Hub or need a quick refresher on how to create an integration, read this quick-start guide for a summary of each step and click the links to learn more.

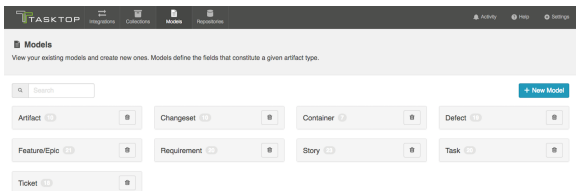
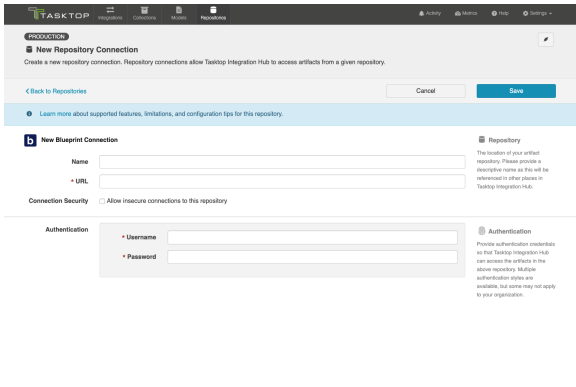
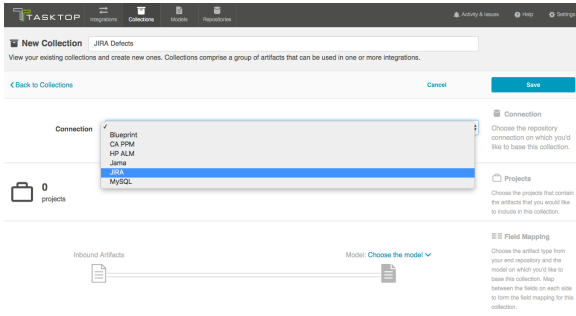
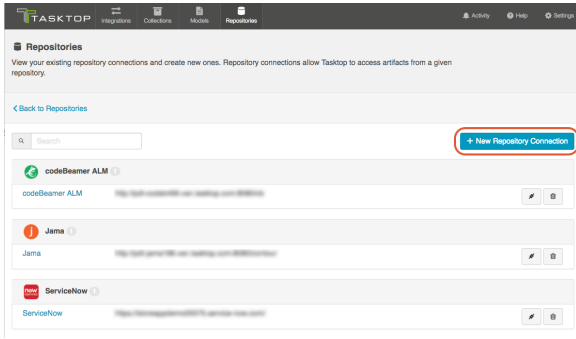
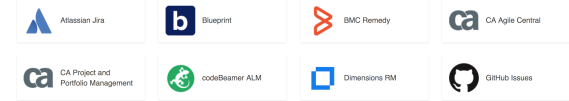
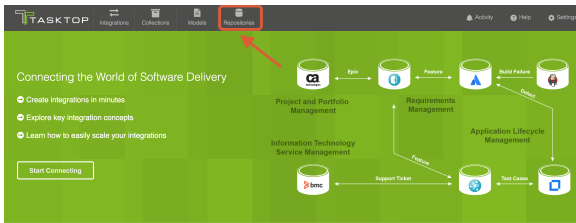
After you have successfully logged into your Tasktop account and configured your settings, you are ready to set-up your integration!

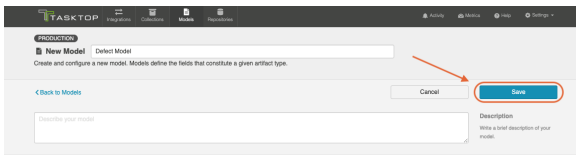
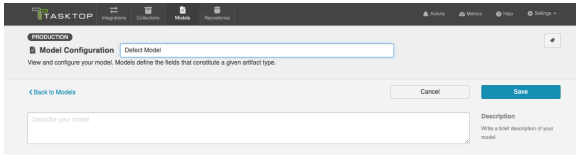
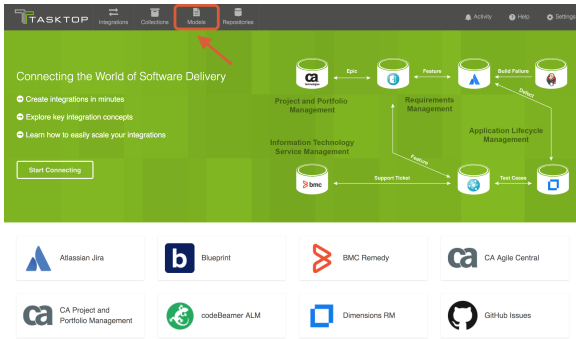
Setting up a new integration takes four simple steps.

1. Connect to your [repository](#).
2. Construct your [model](#).
3. Create your [collection](#) and map it to the model.
4. Configure your [integration](#) using one of our templates.

Finally, once you've configured your integration, you can easily [expand or modify your integration](#).







**Model Configuration** Select Model

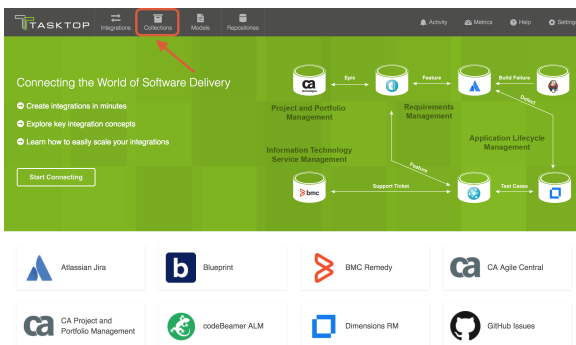
View and configure your model. Models define the fields that constitute a given artifact type.

Model Type: Standard Model

Smart Field	Label	Type	Required
Formatted ID	Formatted ID	String	<input type="checkbox"/>
Summary	Summary	String	<input type="checkbox"/>
Description	Description	String	<input type="checkbox"/>
Priority	Priority	Single Select	<input type="checkbox"/>
Creator	Creator	Person	<input type="checkbox"/>
Modified By	Modified By	Person	<input type="checkbox"/>

**Fields**  
Define the fields and values for this model.

**Priority Values**  
 High  
 Medium  
 Low  
 Allow unmapped values to flow



**TASKTOP** Integrations Collections Models Repositories Activity Help Settings

### Collections

View your existing collections and create new ones. Collections comprise a group of artifacts that can be used in one or more integrations.

Search  [+ New Collection](#)

- Bugzilla**
  - Bugzilla Defects 1 Project Model Requirement
- GitLab**
  - GitLab Issues 1 Project Model Requirement
  - Artifact Type Issue
- ServiceNow Express**
  - ServiceNow Express Incidents 1 Project Model Requirement
  - Artifact Type Incident

**TASKTOP** Integrations Collections Models Repositories Activity Help Settings

### New Collection: Select Collection Type

Select your collection type, then start configuring your new collection. Collections comprise a group of artifacts that can be used in one or more integrations.

[Back to Collections](#)

- Work Item Collection** Access work item artifacts from a given repository.
- Container Collection
- Gateway Collection

**TASKTOP** Integrations Collections Models Repositories Activity Help Settings

### PRODUCTION New Collection Jira Defects

Configure your new collection. Collections comprise a group of artifacts that can be used in one or more integrations.

[Back to Collections](#)

Description:

Repository: **Jira** (Selected)

- Atlassian Jira
- Atlassian Service
- codebunker ALM
- code
- Jira - all versions
- Micro Focus ALM
- MySQL
- Planner
- Test Manager
- ServiceNow
- zendesk 1

Projects:

Inbound Artifacts:  Map Fields

**Manage Projects**

**Available Projects (4)**

Search

- Load Testing Project
- Manual Test Project A
- Manual Test Project B
- Test Project C

Showing 4 of 4 (0 selected)

**Projects in Collection (2)**

Search

- Test Project A
- Test Project B

Showing 2 of 2 (0 selected)

Selected >

All >>

< Selected

<< All

**TASKTOP** Integrations Collections Models Repositories Activity Help Settings

### PRODUCTION New Collection Jira Defects

Configure your new collection. Collections comprise a group of artifacts that can be used in one or more integrations.

[Back to Collections](#)

Description:

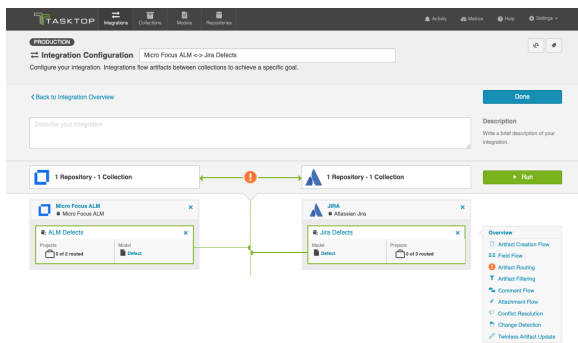
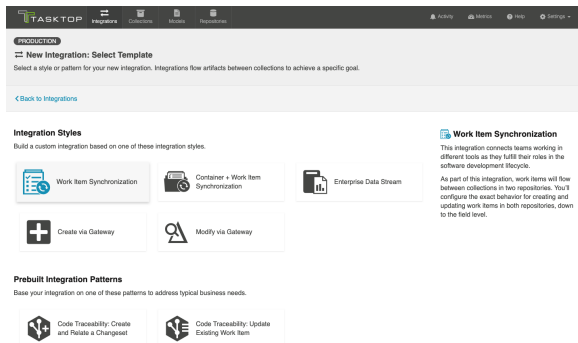
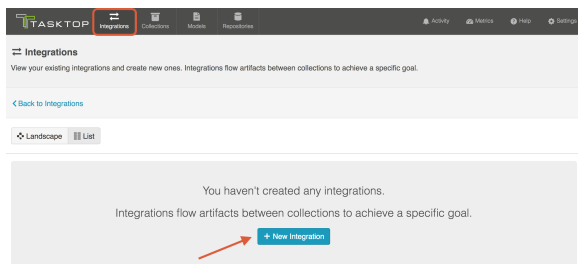
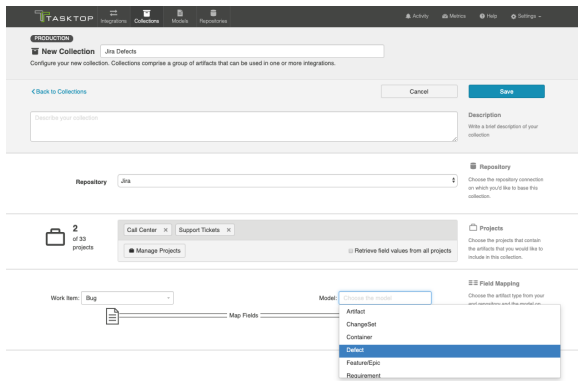
Repository: **Jira**

Projects:

Work Item: **Bug** (Selected)

- Task
- Sub-task
- Story
- Bug
- Epic
- Sprint

Model:





**TASKTOP** Integrations Collections Models Repositories Activity Metrics Help Settings

**PRODUCTION** Integration Configuration Jira Bugs ↔ qTest Manager Defects

Configure your integration. Integrations flow artifacts between collections to achieve a specific goal.

[Back to Integrations](#) **Done**

Describe your integration

Description  
Write a brief description of your integration.

1 Repository - 1 Collection 1 Repository - 1 Collection **Run**

**Jira** # Atlassian Jira

- Jira Defects
- Project: 1 of 1 loaded
- Model: Defect

**qTest Manager** # QCSymphony qTest Manager

- qTest Manager Defects
- Project: 1 of 1 loaded
- Model: Defect

**Overview**

- Affect Creation Flow
- Affect Field Flow
- Affect Hooking
- Affect Filtering
- Affect Comment Flow
- Affect Attachment Flow
- Conflict Resolution
- Change Detection
- Testcases Artifact Update

**TASKTOP** Integrations Collections Models Repositories Activity Metrics Help Settings

**PRODUCTION** Integration Configuration Jira Bugs ↔ qTest Manager Defects

Configure your integration. Integrations flow artifacts between collections to achieve a specific goal.

[Back to Integrations](#) **Done**

Describe your integration

Description  
Write a brief description of your integration.

1 Repository - 1 Collection 1 Repository - 1 Collection **Stop**

**Jira** # Atlassian Jira

- Jira Defects
- Project: 1 of 1 loaded
- Model: Defect

**qTest Manager** # QCSymphony qTest Manager

- qTest Manager Defects
- Project: 1 of 1 loaded
- Model: Defect

**Overview**



- Affect Creation Flow
- Affect Field Flow
- Affect Hooking
- Affect Filtering
- Affect Comment Flow
- Affect Attachment Flow
- Conflict Resolution
- Change Detection
- Testcases Artifact Update


# Step 1: Connect to Your Repository

## Types of Repositories

The first step to take when configuring an integration is to connect to your repository. Your repositories refer to the external tools that Tasktop will flow information between.

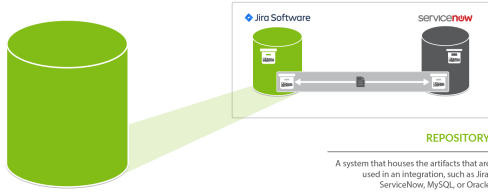
You can create two types of repository connections:

 Standard Repository	 Database Repository
<i>Standard Repositories are available in all Editions.</i>	<i>Database Repositories are only available in Editions that contain the Enterprise Data Stream add-on. See <a href="#">Tasktop Editions table</a> to determine if your edition contains this functionality.</i>
A <b>standard repository</b> refers to an external tool, such as Jira or ServiceNow.  These are software lifecycle tools that contain artifacts, such as defects or requirements.	A <b>database repository</b> refers to an external database, such as MySQL or Oracle.  Database repositories are used as part of the Enterprise Data Stream add-on.
<a href="#">Learn More</a>	<a href="#">Learn More</a>

 **Note:** If you are creating a Gateway collection, for use with our Gateway add-on, no step needs to be taken on the Repository screen.

# Standard Repository Connection

## What is a Repository?



A **repository** is any system that houses the artifacts that can be used in an integration. Repositories can be systems used as part of the software delivery process, like **Micro Focus (HPE ALM)**, **CA Agile Central**, **Jira**, etc., or repositories can be more generic databases, like **MySQL** or **Oracle**.

A **repository connection** is a connection to a specific instance of a given repository that permits **Tasktop** to communicate with that repository. To configure a repository connection, users will need to provide base credentials such as a server URL, a username, and a password.

A **standard repository** is a software lifecycle tool such as Jira or ALM that contains artifacts such as defects or requirements.

## Video Tutorial

Check out the video below to learn how to create a new repository connection:

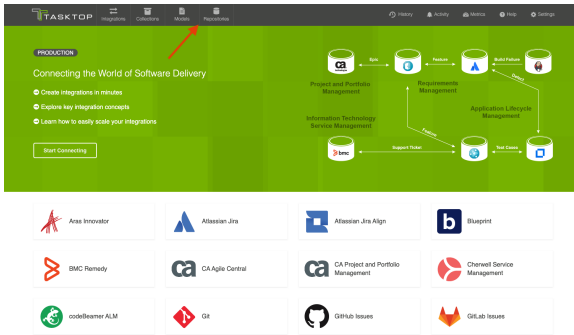
### Before You Begin

1. Review the [User Management](#) section for instructions on how to log in and manage your user accounts.
2. Set a [Master Password](#), which will be used to encrypt your repository credentials.
3. Apply your [License](#) on the Settings screen. You can learn how to apply your license [here](#).

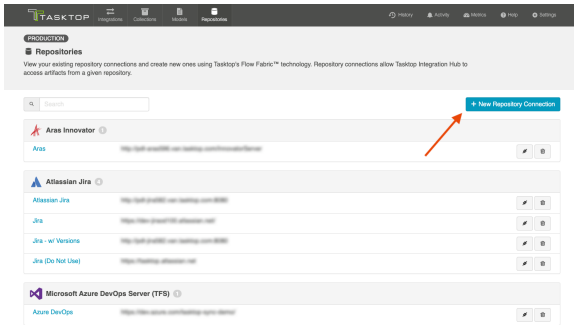
## Connecting to a Standard Repository

### Creating a New Connection

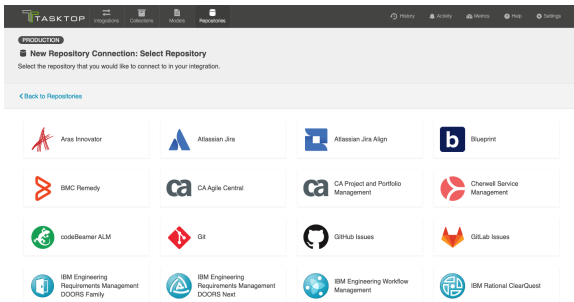
To create a repository connection, select **Repositories** at the top of the screen.



Click + New Repository Connection.

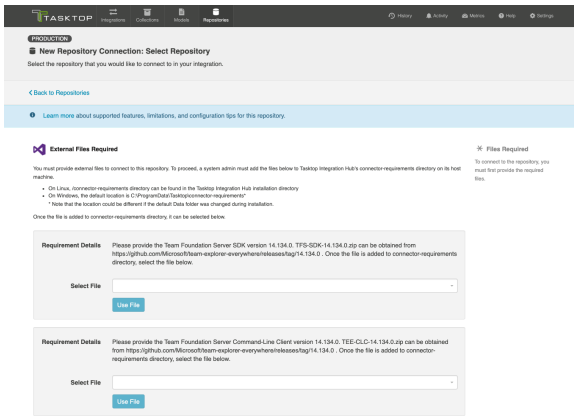


Select the logo of the repository you would like to connect to in your integration.



## Uploading External Files

For certain repositories, such as Microsoft Azure DevOps Server (TFS), external files must be uploaded before navigating to the New Repository Connection screen. If so, you will see a screen similar to the one below. If you do not see this screen, you can disregard this section.



To upload the files, a system administrator (a user with file system access to the machine that hosts Tasktop) must add the files to the designated directory:

- On **Windows**, the default folder is `C:\ProgramData\Tasktop\connector-requirements`
- On **Linux**, the `connector-requirements` can be found in the Tasktop installation directory
- If needed, the user can change the location in which Tasktop looks for the files. This is done by changing the system property `connector.requirements.path`

Once uploaded, select the file from the options available and click **Use File**.

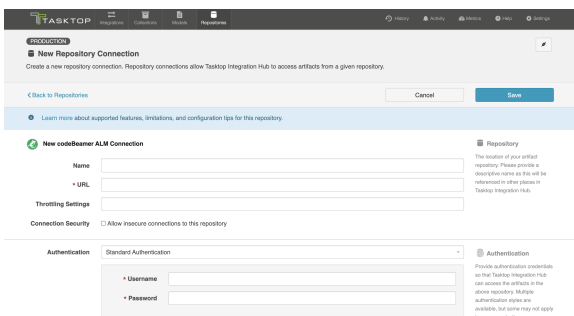
## New Repository Connection Screen

After selecting the repository, the New Repository Connection screen will appear.

To connect to a repository, you must populate the following fields:

- **Name:** This is the name you will give to your Repository Connection. This is how it will be referenced throughout the Tasktop Application.
- **URL:** This is the URL used to access the repository.
- **Authentication Details** (see authentication section below for more details).

**Note:** You may see additional fields on the Repository Connection screen depending on which repository you are connecting to. See our [Connector Documentation](#) for repository-specific information. Any required fields will be marked with an asterisk.



## Throttling Settings

*This functionality should only be used under the guidance of Tasktop support.*

On the Repository Connection screen, you'll see a **Throttling Settings** field. This field can be used to reduce the load placed on a repository when rate limiting or server overload is causing performance delays. It does this by limiting the number of API calls Tasktop processes per minute for the repository. This field will default to blank/not set unless a value is specified.

**⚠ Note:** Caution should be used when setting this value. The ideal value for Throttling Settings is highly dependent on each customer's unique environment. Determining the appropriate value is best achieved through experimentation, using feedback from performance monitoring to tune the value, and making adjustments as necessary. Setting the value too high can potentially cause significant performance degradation when using the Tasktop application.

## Connection Security

On the Repository Connection screen, you will also notice a **Connection Security** checkbox. This will default to unchecked (requiring secure connections). If unchecked, your connection **must** start with HTTPS and have SSL certificate validation enabled. If either condition is not met, Tasktop will **not** connect and provide an error message. If you choose to check the **Allow insecure connections...** checkbox, these restrictions will be lifted.

**⚠ Note:** If allowing insecure connections, please ensure that that configuration aligns with your organization's security policy and the associated risks are understood and accepted.

*Tasktop Cloud users must connect to external repositories via HTTPS.*

Installing a Certificate for HTTPS

To install a certificate for HTTPS, please follow instructions below:

1. Get the public certificate from the third party tool.
2. Copy the certificate to C:\Program Files\Tasktop\jre\lib\security.
3. Stop Tasktop.
4. Open a command prompt and navigate to directory C:\“Program Files”\Tasktop\jre\lib\security.
5. Type "keytool -import -file certfilename -alias reponame -keystore cacerts" (e.g., "keytool -import -file c:\somedir\filename -alias Jira -keystore cacerts").
6. You will be asked for the keystore password which is likely **changeit** unless it has already been changed.
7. You will be asked if you want to trust the certificate to which you reply **y**.
8. You should see a message stating the certificate was imported. If not, something has gone wrong.
9. Start Tasktop.

## Authentication

We recommend that you create a new user within your external tool, to be used only for your Tasktop integration. This is the user information you will enter when setting up your repository connection within Tasktop Hub. By creating a new user, you will ensure that the correct permissions are granted, and allow for traceability of the modifications that are made by the synchronization.

In general, your Tasktop user account should have sufficient permissions to create, read, and update artifacts in your repository. However, depending on the use case, your user may need different permissions. For example, if you are only interested in flowing data **out** of your repository, your user may not need to have full CRUD access, as the **create** and **update** permissions may not be needed.

Please see our [Connector Documentation](#) for repository-specific information regarding user permissions.

Your user should have a secure password. Please be aware that Tasktop will not allow you to save a repository connection utilizing a weak password, such as **tasktop**.

**Note:** For most repositories, you will see a username and password field in the Authentication section. However, some repositories include additional Authentication options.

For most scenarios, you will select **Standard Authentication**. This is where you will enter the username and password used to access the repository. We recommend creating login credentials specifically for Tasktop to access your repository.

## Proxy Server


If Tasktop is installed behind a firewall, you may need to connect to external repositories (e.g. hosted or cloud ALM tools) through a proxy. To create a connection to such external repositories in Tasktop, you can make Tasktop connect through your proxy by configuring the proxy settings when creating a new repository connection. It is recommended to create login credentials specifically for Tasktop on the proxy server.

**Note:** Note that the Proxy Location must be a URL in order for the proxy connection to work. If a .pac script is used in your browser, you will need to open the script and find the URL/port to enter in the Location field.

To use a proxy server, check the **use proxy server** box and fill in your proxy details in the **Proxy Server** section on the New Repository Screen:


## Manual Change Detection

Tasktop's default global change detection settings can be found on the [General \(Settings\)](#) screen. However, if you'd like to override the global defaults, you can configure repository-specific change detection and full scan intervals on this screen.

 Note that you can also set integration-level change detection on the [Change Detection](#) screen.


- The **Change Detection Interval** is the time between polling requests to detect *only changed artifacts*. This defaults to 1 minute on the General (Settings) screen, but can be customized as desired.
- The **Full Scan Interval** is the time between polling requests to detect changed artifacts, in which all artifacts that have previously synchronized in the integration are scanned.

**Note:** For **Hub Cloud instances**, we recommend setting the change detection interval to at least 1 minute and the full scan interval to at least 24 hours.

 When configuring change detection settings, integration-level change detection has the highest precedence, followed by repository-level change detection, and then global change detection. This means that if integration-level change detection is configured, the integration-level setting will **always** be used (even if repository-level or global settings are configured). If no overrides are set, change detection will default to the global settings.

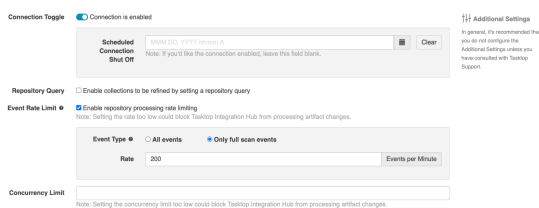
You can learn more about change detection and full scan styles in our FAQ [here](#).

## Additional Settings

 **Note:** In general, we recommend that you do **not** configure the Additional Settings unless you have consulted with Tasktop Support.

Additional settings you can configure on the Repository Connection screen include:

- Connection Toggle
- Repository Query
- Event Rate Limit
- Concurrency Limit



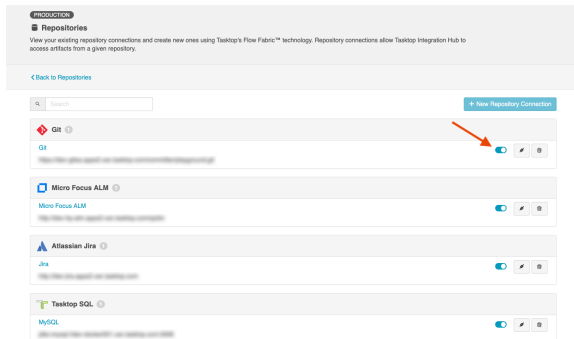
The **Connection Toggle** can be used to disable individual repository connections, stopping all traffic going into a repository. Additionally, you can use this option to schedule your repository connections to be disabled during mandatory credential updates to prevent user lockout.

Learn more in the section [below](#).



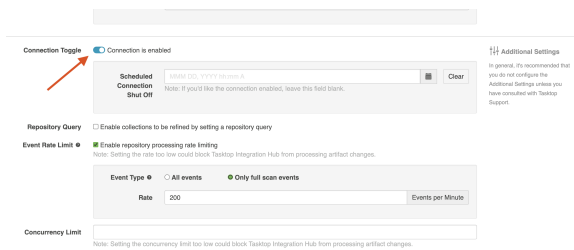
# Toggle your Repository Connection

To enable or disable a repository connection, click the toggle on the **Repositories** screen or on the **Repository Connection** screen.



On the Repository Connection screen, you can also schedule a repository connection to be disabled on a certain date or time.

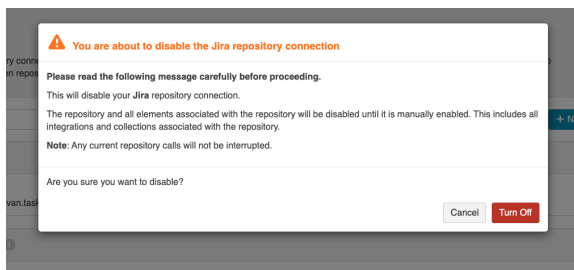
**Note:** The entered date/time should be in the following format MMM DD, YYYY hh:mm A (e.g., Apr 01, 2022 05:00 PM)



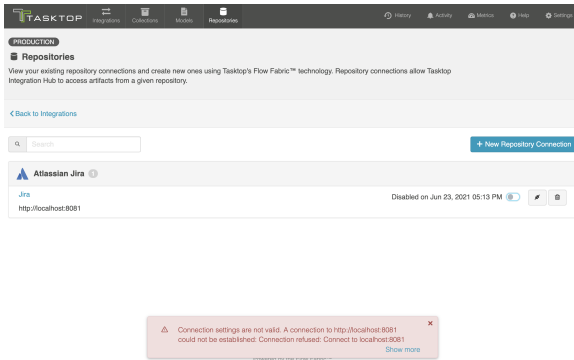
If you choose to disable a repository connection, a pop-up will appear confirming you'd like to proceed.

Once confirmed, all elements associated with the repository connection will be disabled until the connection is enabled.

**Note:** This pop-up will not appear if scheduling the repository connection to be disabled.

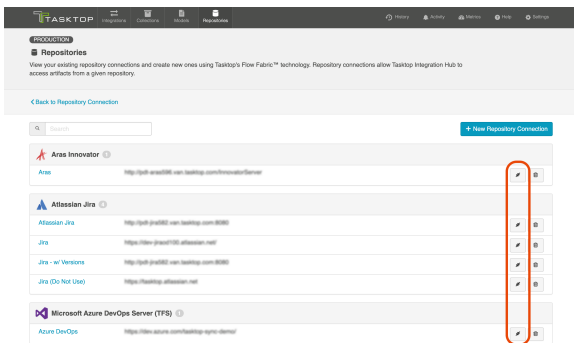
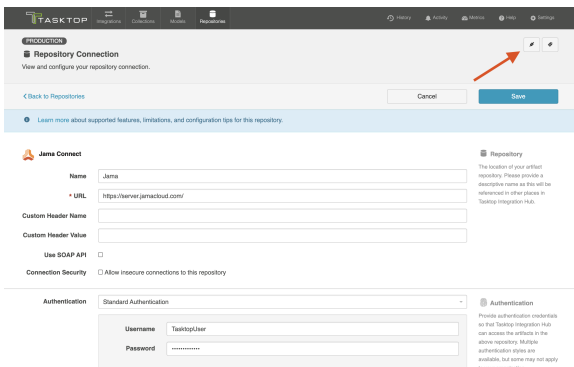


Once the connection has been disabled, the repository must be up and running to enable the connection again. If not, you'll see an error message at the bottom of the screen.

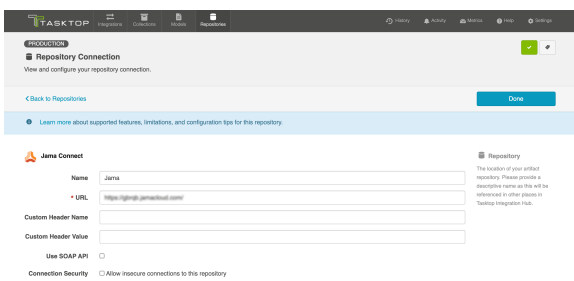


## Testing your Repository Connection

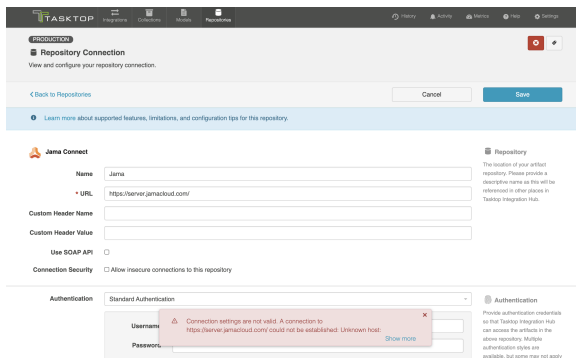
To test your repository connection, click the **Test Connection** button on the Repository Connection screen, or click the icon on the Repositories screen.



You will see a success or failure message to confirm whether Tasktop was able to connect to your repository.

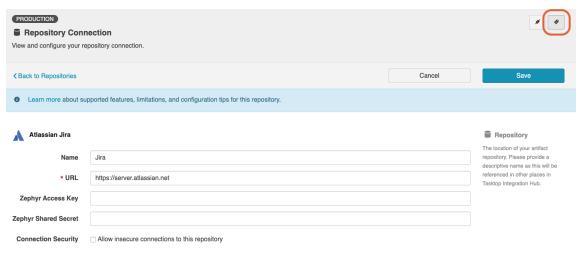


When your repository fails to connect, you will also see an error message at the bottom of the screen with additional details on the source of the failure.



## Viewing Associated Configuration Elements

To view associated configuration elements (such as collections or integrations that utilize the repository connection you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



### Associated Elements for Repository Connection "Jira"

**4 Integrations using this Repository Connection**

- ALM <-> Jira Defects
- Defect Reporting
- Jira Defect Creation
- Jira Stories <-> ALM Requirements

**3 Repository Collections using this Repository Connection**

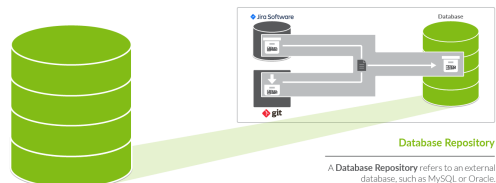
- Jira Defects
- Jira Stories
- Jira Stories B

Close

# Database Repository Connection

## What is a Database Repository Connection?

*Database Connections are only available in Editions that contain the Enterprise Data Stream add-on. See [Tashtop Editions table](#) to determine if your edition contains this functionality.*



A **database repository**, is a tool such as MySQL or Oracle, which allows you to flow data to a central database. Database repositories are used as part of the Enterprise Data Stream add-on.

In order to configure an Enterprise Data Stream Integration, you must first connect to the database that will be used by that integration. Creating a new database connection is similar to creating a [standard repository connection](#), with a few extra considerations. To create a new database connection, follow the steps below.

## Before You Begin

1. Review the [User Management](#) section for instructions on how to log in and manage your user accounts.
2. Set a [Master Password](#), which will be used to encrypt your repository credentials.
3. Apply your [License](#) on the Settings screen. You can learn how to apply your license [here](#).

## Supported Databases

The following databases and versions are supported for use with the Enterprise Data Stream add-on:

### PostgreSQL

#### General Support

- 10
- 11
- 12
- 13
- 14

## Extended Support

- 9.6 (*End-of-Service-Life Date: 18 Jan 2023*)

💡 If you are interested in extended support, please reach out to your [Tasktop contact](#).

## Microsoft SQL Server

### General Support

- 2017
- 2019

### Extended Support

- N/A

## Oracle

### General Support

- 19c
- 21c

### Extended Support

- 18c (*End-of-Service-Life Date: 18 Jan 2023*)

## MySQL

We recommend using JDBC driver version 8.0 or later when creating a SQL connection for Enterprise Data Stream integrations.

### General Support

- 5.7
- 8.0

### Extended Support

- N/A

💡 **Note:** The user must be a SQL authenticated user (and not a Windows authenticated user)

## Database Connections and Encryption

The following section describes different ways to configure your database connection. If you choose not to encrypt your connection, data will be transmitted over the network unprotected and will be at

risk of being intercepted. Likewise use of self-signed certificates or other certificates not signed by a trusted Certificate Authority puts your data at risk as Tasktop cannot verify the identity of the server at the end of the connection.

Please ensure your connection is configured in a way that is aligned with your security policy and the associated risks are understood and accepted.

## Configuration Details

### PostgreSQL

or PostgreSQL, please refer to [PostgreSQL documentation](#) for more information.

#### Location

- Example Format: `jdbc:postgresql://hostServerName:postgresServerPort/MyDatabaseName`

You can enable encrypted connections by setting 'ssl=true' (e.g., `jdbc:postgresql://<server-name>:<port>/?ssl=true`).

If the certificate for the PostgreSQL server is self-signed you'll need to set 'sslfactory=org.postgresql.ssl.NonValidatingFactory' and 'sslmode=require' (e.g., `jdbc:postgresql://<server-name>:<port>/?ssl=true&sslmode=require&sslfactory=org.postgresql.ssl.NonValidatingFactory`).

If the certificate for the PostgreSQL server is not self-signed you'll need to add the certificate to the JDBC's [truststore](#).

### Microsoft SQL Server


For SQL Server, please refer to [Microsoft documentation](#) for more information.

#### Location

- Example Format: `jdbc:sqlserver://hostServerName;instanceName=MyInstance;datasenname=MyDatabaseName`

You can enable encrypted connections by setting 'encrypt=true' (e.g., `jdbc:sqlserver://<server-name>:1433;encrypt=true;trustServerCertificate=false`). If the certificate for the MySQL server is self-signed you'll need to set 'trustServerCertificate=true' (e.g., `jdbc:sqlserver://<server-name>:1433;encrypt=true;trustServerCertificate=true`)

If using JDBC driver `mssql-jdbc-10.2.x` or later, the `trustServerCertificate="` parameter and its corresponding value is required by the driver.

 **Note:** Some older editions may be missing security updates and will need to [apply security service packs](#) to use a self-signed certificate and encryption. You may experience certificate errors if the SQL Server is using a self-signed or corporate certificate. To work around this, you will need to disable certificate validation in the JDBC driver or add the certificate to the JDBC's truststore.

## Oracle

For Oracle, please refer to this whitepaper for an overview of how to set up connections to encrypted Oracle server. For a guide to configuring the Oracle server to support SSL, please refer to [Oracle documentation](#).


### Location

- Example Format: `jdbc:oracle:thin:@hostServerName:oracleServerPort/SID`

For the most part assuming that the server is set up properly, you can follow Case#2 in the white paper and simply use a URL with the following format: `jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(PORT=2484)(HOST=<hostname>))(CONNECT_DATA=(SERVICE_NAME=<servicename>)))`. On the server, make sure to disable client authentication by setting 'SSL\_CLIENT\_AUTHENTICATION=FALSE ' in the listener.ora and sqlnet.ora files.

For unencrypted connections, the protocol should be **TCP** and the port would generally be 1521, but the URL would otherwise be the same. The above example connection string is formatted in the 'Oracle Net connection descriptor' format, but Tasktop also accepts 'Thin-style service name' connection strings such as `jdbc:oracle:thin:@<hostname>:1521:<servicename>`.

If the certificate for the Oracle server is self-signed, but you still want to use SSL, you will need to follow Case#1 in the white paper. As described in the paper, only anonymous cipher suites are permitted when trying to use SSL without server authentication. You can specify the cipher suites in the sqlnet.ora file on the Oracle server.

 **Note:** Some versions of Oracle do not by default support anonymous cipher suites. Thus, they will need to be imported to the server before enabling them.

## MySQL

For MySQL, refer to [MySQL documentation](#) for the details on how to set up your connection.

### Location

- Example Format: `jdbc:mysql://hostServerName:mysqlServerPort/MyDatabaseName`

To enable encryption on older MySQL servers (5.6.25 and earlier or 5.7.5 and earlier) you need to set the connection property 'useSSL=true' (e.g., `jdbc:mysql://<server-name>:3306?useSSL=true`). Later versions will implicitly try to connect using an encrypted connection. Regardless of the version, the client will only enforce that the server uses TLS if the property 'requireSSL=true' is set.

If the certificate for the MySQL server is self-signed you will need to set 'verifyServerCertificate=false' (e.g., `jdbc:mysql://<server-name>:3306?useSSL=true&verifyServerCertificate=false`).

## Step 1: Download the JDBC Driver

## Microsoft SQL Server

The JDBC driver for Microsoft SQL Server can be downloaded from the [Microsoft support site](#). The SQL Driver Location should reference the directory containing the `sqljdbc42.jar` file. This file should be the only .jar file in that directory, or you may end up with errors upon configuring your collection.

Tasktop currently supports use of the 7.0.0.jre8 driver version.

## MySQL

The JDBC driver for MySQL can be downloaded from the [MySQL download site](#). The SQL Driver Location should reference the directory containing the `mysql-connector-java-<version>-bin.jar` file.

## Oracle

The JDBC driver for Oracle can be downloaded from the [Oracle support site](#). Note that it is best if the Oracle JDBC driver that is used matches the version of the Oracle server that you are connecting to. Additionally, the `ojdbc6.jar` file is the only file that should be in the directory that is used for the SQL Driver Location or you may end up with errors upon configuring your collection.

## PostgreSQL

The JDBC driver for PostgreSQL can be downloaded from the [PostgreSQL download site](#). The SQL Driver Location should reference the directory containing the `postgresql-<version>.jar` file.

# Step 2: Upload the JDBC driver

The SQL driver files must be put on the file system of the same server where Tasktop is installed. When setting up a connection to your database with the SQL connector, the SQL Driver Location field should reference the location of the SQL driver files on the server.

## Microsoft SQL Server

The SQL Driver Location should reference the directory containing the `sqljdbc42.jar` file. This file should be the only .jar file in that directory, or you may end up with errors upon configuring your collection.

## MySQL

The SQL Driver Location should reference the directory containing the `mysql-connector-java-<version>-bin.jar` file.

## Oracle

The SQL Driver Location should reference the directory containing the `ojdbc6.jar` file. The `ojdbc6.jar` file should be the only file in that directory, or you may end up with errors upon configuring your



collection. Note that it is best if the Oracle JDBC driver that is used matches the version of the Oracle server that you are connecting to.

## PostgreSQL

The SQL Driver Location should reference the directory containing the `postgresql-<version>.jar` file.

## Step 3: Connect to your Database

1. In Tasktop, click **Repositories** at the top of the screen, and click **New Repository Connection**.
2. Select 'Tasktop SQL' as the repository.
3. Enter a label for your connection. This is how it will be referenced through the Tasktop application.
4. Enter the URL of your database. The protocol should be "jdbc:sqlserver://" for a MS SQL database, "jdbc:mysql://" for a MySQL database, "jdbc:oracle://" for an Oracle database, or "jdbc:postgresql://" for a PostgreSQL database.
5. Select the appropriate JDBC driver (SQL Server, MySQL, Oracle, or PostgreSQL).
6. Enter the SQL driver location, which is the location of the SQL driver files on the Tasktop server. See steps 1 and 2 above for more information on the SQL driver files.
7. Enter a username and password for your database.
8. If you'd like, you can test your connection by clicking the **Test Connection** button in the upper right corner.
9. In general, we recommend that users do not edit the Concurrency Limit or Event Rate Limit fields. You can learn more about these fields [here](#).
10. Click **Save** and then **Done** to save the connection.

The screenshot shows the 'New Repository Connection' form in the Tasktop interface. The form is divided into three main sections: 'New Tasktop SQL Connection', 'Repository', and 'Authentication'. The 'New Tasktop SQL Connection' section contains fields for 'Name' (MySQL), 'JDBC URL' (jdbc:mysql://mysql.server.tasktop.com:3306), 'JDBC Driver', and 'SQL Driver Location'. The 'Repository' section has a dropdown menu. The 'Authentication' section has fields for 'Username' (TasktopUser) and 'Password'. There are 'Cancel' and 'Save' buttons at the top right of the form. A warning message is displayed above the form, stating that Tasktop Integration Hub does not enforce the security of the connection and that users should ensure encryption and security settings are aligned with their company's security policy.

## Viewing Associated Configuration Elements

To view associated configuration elements (such as collections or integrations that utilize the repository connection you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.

**PRODUCTION**

**Repository Connection**  
View and configure your repository connection.

[Back to Repositories](#)

Tasktop Integration Hub does not enforce the security of the connection. Please ensure the encryption and security settings are aligned with your company's security policy. See our [user docs](#) for more details.

---

**Tasktop SQL**

Name

URL

\* JDBC Driver

SQL Driver Location

**Authentication**

Username

Password

**Repository**  
The location of your artifact repository. These provide a descriptive name as this will be referenced in other places in Tasktop Integration Hub.

**Authentication**  
Provide authentication credentials so that Tasktop Integration Hub can access the artifacts in the above repository. Multiple authentication types are available, but some may not apply to your repository.

## Associated Elements for Repository Connection "MySQL"

### 1 Integration using this Repository Connection

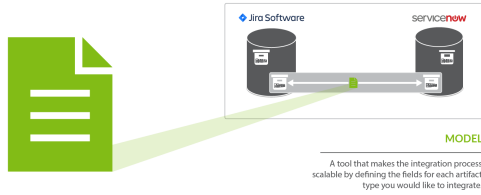
- [Defect Reporting](#)

### 1 Repository Collection using this Repository Connection

- [Mysql Defects](#)

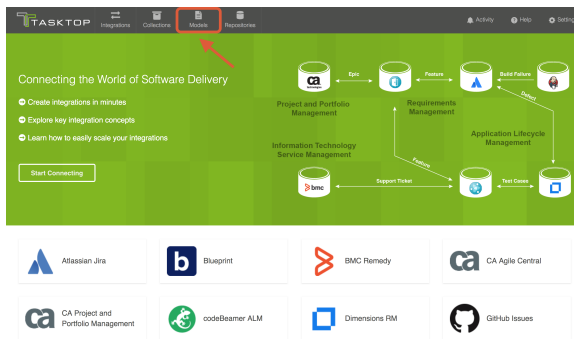
# Step 2: Create or Reuse a Model

## What is a Model?



A **model** is a tool that makes the integration process scalable by defining the fields for each artifact type you would like to integrate. By mapping collections to the same model, you will be able to easily add new repositories and new projects within those repositories to your integration landscape. You can learn more about models in the [Key Concepts](#).

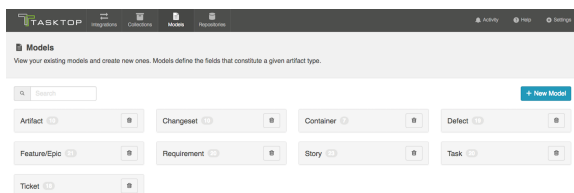
To access your models, click on the **Models** button at the top of the screen.



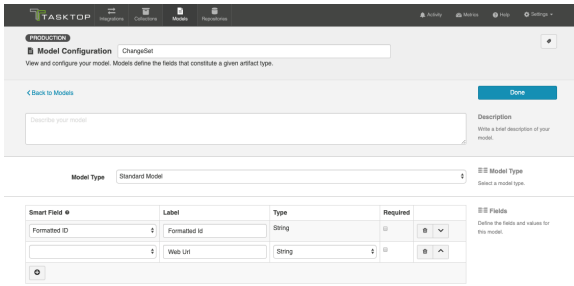
## Out of the Box Models

Tasktop comes pre-packaged with several out-of-the-box models that are ready for you to use!

On the Models screen, you will see the name of each model, with a number identifying how many fields are included in that model.

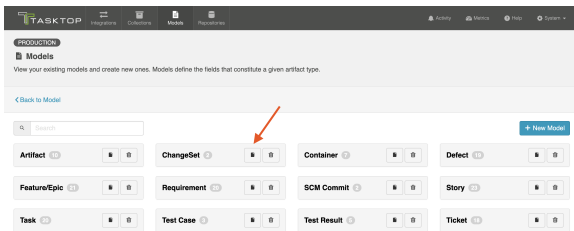


To view a model, simply click on its title. You will be brought to the Model Configuration screen, which will show the fields included in that model.



## Copying a Model

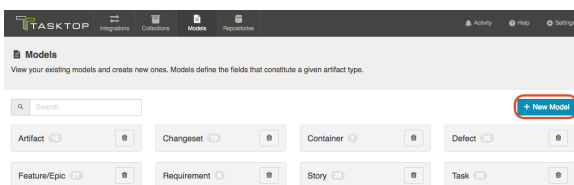
There may be times when you wish to copy a model and then modify certain fields on it for use in a different integration. To copy a model, click the **copy** button from the Models screen. The copied Model will be named "<Original Model name> (copy)".



## Custom Models

Check out the video below to learn how to create a new custom model:

To create a new custom model, click the **+ New Model** button at the top of the screen.

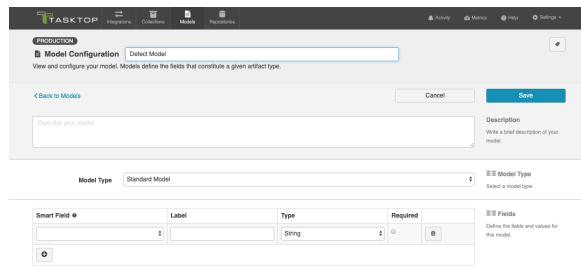


## Model Type

Depending on your [Tasktop edition](#), you may see a **Model Type** dropdown. You can learn more about this on the [Test Synchronization](#) page.

## Add Fields to Your Model

You can start configuring your first model immediately — just name it and start entering metadata into the first line. To add additional fields to your model, simply click on the plus sign at the bottom left of the model box.



## Smart Field Designation

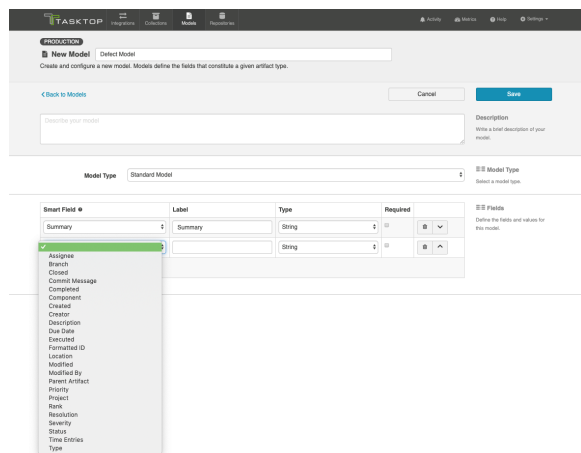
For each field you add to your model, you have the option of identifying its corresponding smart field type. *Smart fields* are a set of fields commonly available in the connectors for all of the repositories Tasktop connects to. By designating a smart field to your model field, Tasktop will be able to more easily match fields from your repositories to your models when you are creating and editing collections.

Selecting a Smart Field will also give Tasktop the power to suggest the proper field type for your model field.

You do not have to select a smart field for all model fields. If you cannot find a smart field that corresponds to a model field, just leave the smart field drop down empty for that field.

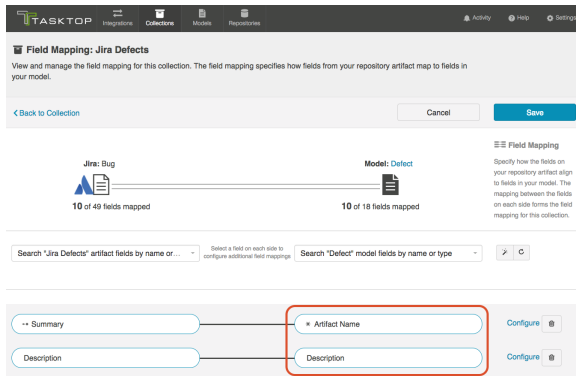
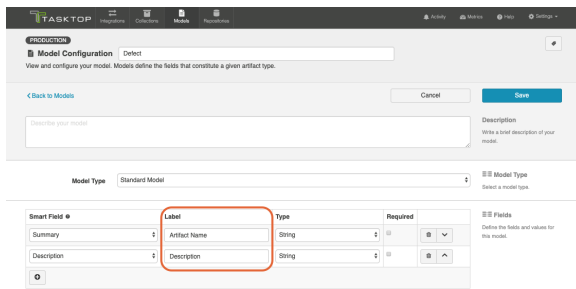
Some examples of smart fields are:

- **Formatted ID:** the human-readable ID of an artifact
- **Location:** the field that holds the URL of an artifact
- **Modified:** a date-time field showing when changes were last made to an artifact



## Field Label

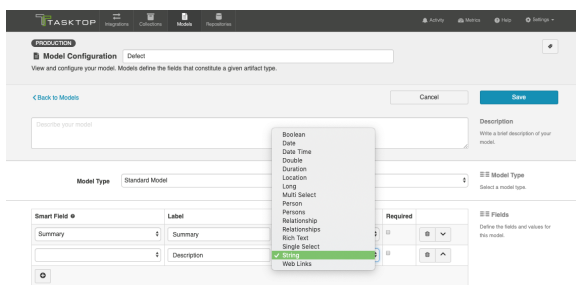
The **label** is the name of the field in your model that you will see throughout the Tasktop application, from the collection-to-model field mapping screen to the Field Flow screen in an integration.



## Field Type

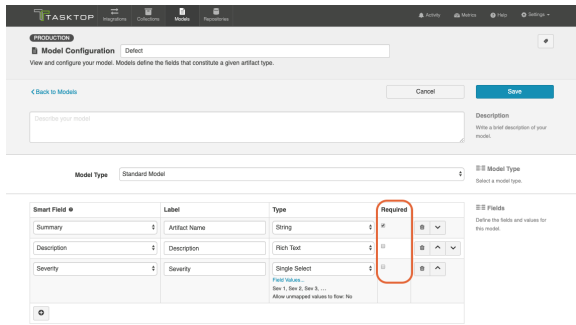
Tasktop supports a number of field types, such as *string*, *multi-select*, *relationship*, and more, for use in your model. Identify the field type that most closely aligns with the type of information you expect to flow through this model field.

Review the sections below for best practices and additional configuration steps for each field type.



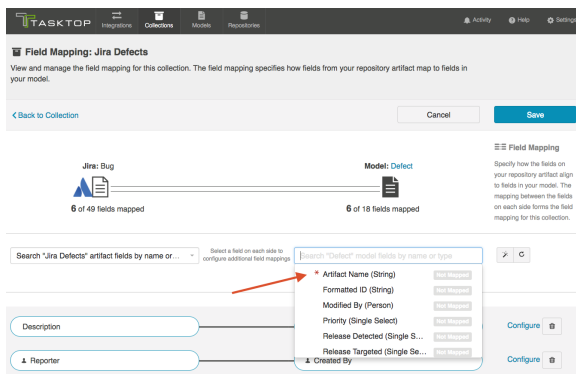
## Required Designation

For each field, you can configure whether or not that field requires a value.



Marking a field as required has implications for all collection types:

- For repository collections, any required model field will be shown with a red asterisk in the collection to model mapping.



- For gateway collections, you will need to pass in a value in the payload for any required field in order for Tasktop to accept the payload.
- For database collections, the suggested DDL will mark the field as required ("not null"); this means that if you use that suggested DDL to create your database tables, the field will be required by your database table to create a new record about an artifact.

## Data Description Language Generator

Database

Model

Suggested DDL

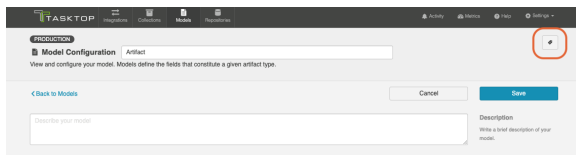
```
CREATE TABLE DEFECT (  
  ID BIGINT (19) AUTO_INCREMENT,  
  FORMATTED_ID VARCHAR (1000),  
  ARTIFACT_NAME VARCHAR (1000) NOT NULL,  
  DESCRIPTION VARCHAR (1000),  
  SEVERITY VARCHAR (255),  
  PRIORITY VARCHAR (255),  
  STATUS VARCHAR (255),  
  RESOLUTION VARCHAR (255),  
  RELEASE_DETECTED VARCHAR (255),  
  SPRINT_DETECTED VARCHAR (255),  
  RELEASE_TARGETED VARCHAR (255),  
  SPRINT_TARGETED VARCHAR (255),  
  CREATED_BY VARCHAR (64),  
  MODIFIED_BY VARCHAR (64),  
  OWNER VARCHAR (64)
```

Execute the DDL and Close to refresh the list of tables.

Close

## Viewing Associated Configuration Elements

To view associated configuration elements (such as collections or integrations that utilize the model you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



Model Type: Standard Model

Smart Field	Label	Type	Required	
Formatted ID	Formatted ID	String	*	
Project	Project	Single Select Field Name: Allow unmapped values to flow: Yes	*	
Type	Type	Single Select Field Name: Allow unmapped values to flow: Yes	*	

### Associated Elements for Model "Artifact"

- 1 Repository Collection using this Model
  - Jira Versions

Close

## Best Practices for Models



- Generally, the fewer models, the better. Create one model per primary artifact type. The model should have the greatest number of fields needed to accommodate all of your integrations for that artifact type. Then, at the collection- and integration-level, you can configure your field flow to only flow whichever fields are relevant for that integration. By utilizing fewer models, you'll see benefits in improved governance and standardization, and greater ease of scalability, data collection, self-service, and maintenance.
- The model field, by definition, sits in the middle of two fields: one from each repository you are integrating. Those two fields in your end systems may have different levels of detail, but by definition, they must map to the same model field. We recommend that your model field match the 'richer' of your two fields. This will ensure you preserve as much information as possible for as long as possible in your integrations. This allows your model to be more reusable and to support more scenarios.

For example, when mapping between text fields, it's often good practice to use a rich text field in your model. That way, you preserve the rich text from the source. If you map a rich text field to a text (string) field in the model, you'll lose the formatting information immediately.

- If you are mapping a single- or multi-select field in your repository that contains a large look-up list (i.e., which has hundreds or thousands of possible values):
  - If the list of values match between your source and target repositories, make the model field a string field. This will allow the values to flow between the repositories without the need to maintain a field mapping.
  - If you only need to map a small sub-set of the values, make the model field a single- or multi-select field, and check **Allow unmapped values to flow**.
- Whenever possible, utilize the smart fields available. For example, if you would like to add a 'status' field to your model, use the 'status' smart field, rather than entering 'status' as the field label, and selecting a field type manually. This will enable Tasktop to auto-map the model field to the appropriate fields within each repository.
- If you would like to use a field for artifact filtering, make sure to include that field in your model.

## Glossary of Field Types

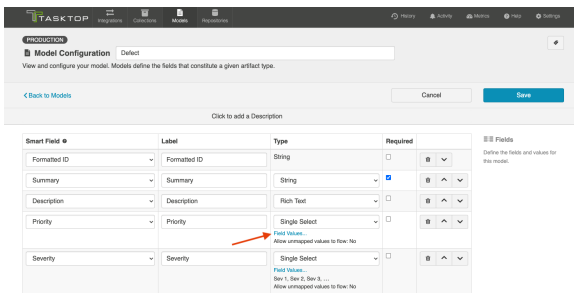
### Fields that Require Additional Configuration

#### Single-Select and Multi-Select

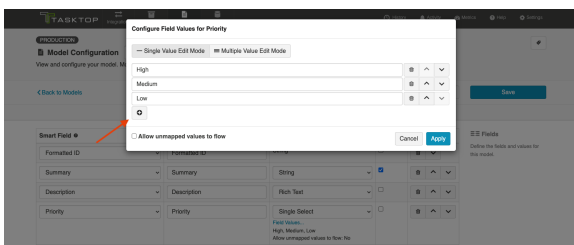
Single-selects and multi-selects fields refer to fields in which the user selects one or many options from a list of values. These fields could refer to drop down menus, checkboxes, or radio buttons within the end repository, to name a few examples.

When utilizing single-select and multi-select fields in your model, there are a couple of additional configuration steps to be aware of.

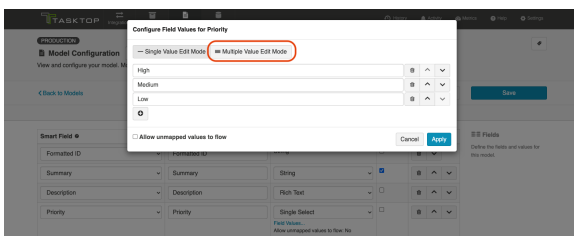
First, click the **Field Values** link to add values to your model. These will be the available field values that you will then map to fields within each end repository.



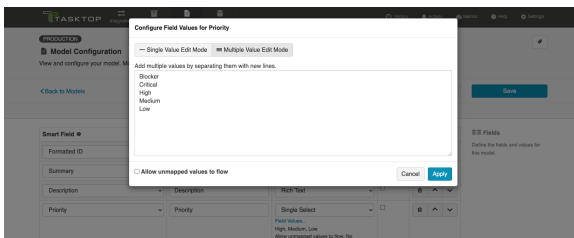
After clicking the link, a pop-up will appear prompting you to configure your field values. If you'd like to add a single field value to your model, you can use the + button to do so.



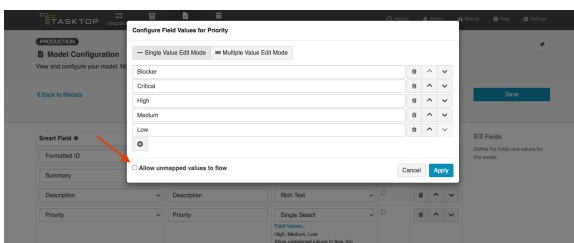
If you'd like to add multiple values at once, select **Multiple Value Edit Mode**.



Here, you can input multiple values by separating each value with a new line.

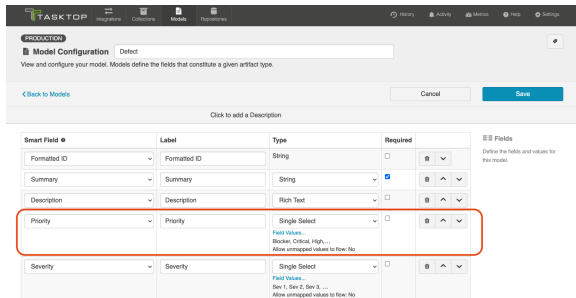


Next, decide whether or not you'd like to allow unmapped values to flow by checking this box.



If you **do not allow unmapped values to flow** (the default setting), the server will reject any value that is not specified in the model. In general, this is the recommended approach. If you select this approach, you will need to map all possible values for the repository field to the specific values for the model field on the Field Configuration screen during Collection configuration.

If you **do allow unmapped values to flow**, field values not specified in the model will be able to flow while the integration is running. This can make sense in a few specific scenarios, such as an Enterprise Data Stream integration or in single select to string transforms, where there are many options available and you don't desire any normalization of the data flowing through. In most cases, however, you will not want to allow unmapped values to flow.



In the image above, you have added 5 specific values for the field "Priority" but have not allowed unmapped values to flow, meaning that any field values sent from the collection will need to be mapped to these 5 model values in order for your artifact to flow successfully.

## Fields that Do Not Require Additional Configuration

### Boolean

Boolean fields are typically represented by checkboxes in the end repository. These fields are often useful for filtering integrations. As an example, you could create a custom boolean field titled "Participate in Tasktop Integration". If you filter by that field (on the [Artifact Filtering](#) screen of your integration), only artifacts that your users have checked will participate in the integration.

### Date

These identify a specific date.

### Date Time

These are fields that identify something more specific than a date. For example, January 1, 2017 9:35am. A 'Created' field is often a Date Time field.

### Double

Use this field for number fields - either integers or decimals. For example, a double could include both values "2" and "2.5." The *Long* field type can also be used for integers.

### Duration

This field holds a length of time. This is typically used for worklogs and time estimations on tasks.

## Location

This model field holds a URL.

There is also a Smart Field called Location which is specifically for the URL of a given artifact. The Location Smart Field is often used when you want to [synchronize a URL reference field to your target artifact](#) (sometimes referred to as 'backlinking'). This allows for bi-directional traceability. It can also be used to report the location of an artifact in an [Enterprise Data Stream integration](#).

The 'Location' *model field type*, on the other hand, can be for any URL.

In addition to 'Location,' you will also see that there is a 'Web Links' field type available. The 'Web Links' field type includes the URL as well as additional information such as label, creator, and time of creation (depending on what the repository supports), while 'Location' includes only the URL.

## Long

This field is for integer or whole numbers, only. An example of a *Long* field value is "2," but *not* "2.5." The *Double* field type can be used if you will also need to cover decimal values. Story points are a good example of a *Long* field.

## Person and Person(s)

You'll notice that you are able to create both 'person' and 'persons' field types in your model. 'Person' refers to fields that contain one, and only one, Person object. Examples of this type of field are: Assignee, Owner, Reviewer, etc. Person objects contain more information than just the display name of the person. For example, they may also utilize the user's e-mail address or username in order to reconcile 'persons' between different repositories. You can learn more about person reconciliation strategies [here](#).

The Person(s) field type refers to fields that contain more than one Person. A 'Watchers' field is a good example. There can be one or more Persons in a single Watchers field.

💡 In general, we recommend using the 'persons' field type in your model, rather than 'person,' especially in cases where you may want to map a 'person' field in one repository to a 'persons' field in your other repository.

## Relationship and Relationship(s)

You'll notice that you are able to create both 'relationship' and 'relationships' field types in your model. 'Relationship' refers to scenarios where your artifact can be related to one, and only, one artifact. An example of a 'relationship,' is 'parent,' as oftentimes an artifact can only have one parent artifact. 'Relationships' refers to scenarios where your artifact can be related to many artifacts. An example of 'relationships' is 'child,' as one parent-artifact can often have many child artifacts.

💡 In general, we recommend using the 'relationships' field type in your model, rather than 'relationship,' especially in cases where you may want to map a 'relationship' field in one repository to a 'relationships' field in your other repository.

## Rich Text

This is for fields that can contain rich text. These are fields that can contain html and/or wiki markup, such as bold, italics, or colored fonts. These are often Description fields.

## String

String fields are used for text input. These model fields will not transmit rich text information.

## Web Links

*Web Links* fields are intended to point to URLs outside of a given tool. They can contain information in addition to the URL, such as label, time of creation, and creator (depending on what the repository supports). They could also be considered a hyperlink field.






In addition to 'Web Links,' you will also see that there is a 'Location' field type available. The 'Web Links' field type includes the URL as well as additional information such as label, creator, and time of creation (depending on what the repository supports), while 'Location' includes only the URL.

# Step 3: Create Your Collection(s)

## Types of Collections

Your collections define which artifacts are eligible to flow as part of your integration.

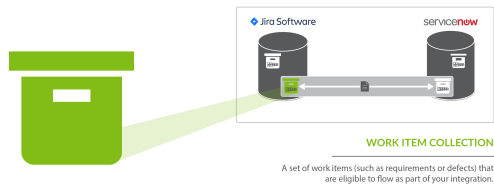
You can create five types of collections:

 <p><b>Work Item Collection (Repository)</b></p>	 <p><b>Container Collection (Repository)</b></p>	 <p><b>Work Item Collection (Database)</b></p>	 <p><b>Gateway Collection</b></p>	 <p><b>Outbound Only Collection</b></p>
<p><i>Work Item Collections (Repository) are available in all Editions.</i></p>	<p><i>Container Collections (Repository) are available in all Editions.</i></p>	<p><i>Work Item Collections (Database) are only available in Editions that contain the Enterprise Data Stream add-on. See <a href="#">Tasktop Editions table</a> to determine if your edition contains this functionality.</i></p>	<p><i>Gateway Collections are only available in Editions that contain the Gateway add-on. See <a href="#">Tasktop Editions table</a> to determine if your edition contains this functionality.</i></p>	<p><i>Outbound Only Collections are only available in editions that have access to the Git repository.</i></p>
<p>A <b>work item collection (repository)</b> contains work items, such as defects or requirements,</p>	<p>A <b>container collection</b> contains containers, such as folders or modules, from</p>	<p>A <b>work item (database) collection</b> connects to a database, such as MySQL or Oracle.</p>	<p>A <b>gateway collection</b> contains artifacts sent via an inbound webhook, from an external tool.</p>	<p>An <b>outbound only collection</b> contains artifacts like code commits or changesets, which you may want to flow out of your repository, but</p>

from repositories, such as Jira or ServiceNow.	repositories such as DOORS Next Generation or Jama.			which would not receive updates into y our repository. <u>    </u>
<a href="#">Learn More</a>	<a href="#">Learn More</a>	<a href="#">Learn More</a>	<a href="#">Learn More</a>	<a href="#">Learn More</a>

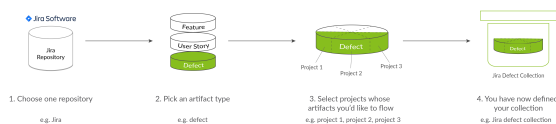
# Work Item Collection (Repository)

## What is a Collection?



You can think of a *collection* as the set of artifacts that are eligible to flow as part of your integration. The process of creating a collection consists of a few steps which whittle down your repository into a smaller subset of artifacts. To create your collection, you will specify:

1. The repository the artifacts live in
  - a. Each collection can only come from *one* repository
2. The artifact type (i.e. defect, requirement, test case, etc)
  - a. Each collection can only contain *one* artifact type
3. The projects within the repository that those artifacts live in
  - a. Each collection can contain one or more projects
4. The model you would like your collection to be mapped to (not pictured)
  - a. Each collection can be mapped to one and only one model



You can learn more about collections in the [Key Concepts](#).

## Types of Work Item Collections

There are two types of Work Item Collections:

- Work Item (Repository) Collections, which connect to repositories like **Jira**, **Jama**, and **ServiceNow**
- [Work Item \(Database\) Collections](#), which connect to databases, such as **MySQL**.

On this page, we will be teaching you how to configure a Work Item (Repository) Collection.

**Note:** SCM repositories, such as Git, are not available for Work Item collections. To configure an SCM collection, please see [Outbound Only Collection](#) instructions.



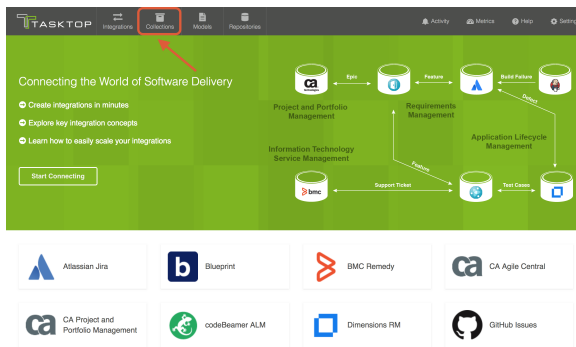
# Video Tutorial

Check out the video below to learn how to create a new work item (repository) collection:

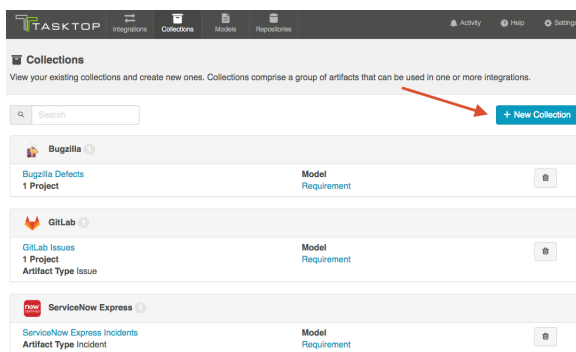
## Creating a Work Item (Repository) Collection

To create a work item (repository) collection, follow the steps below:

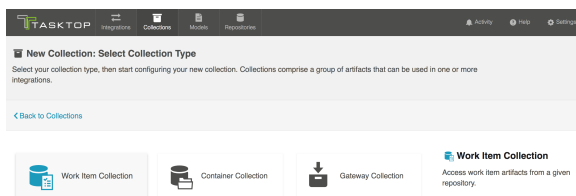
Select **Collections** at the top of the screen.



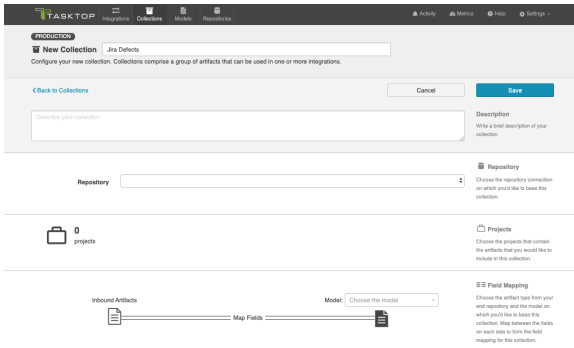
Click **New Collection**.



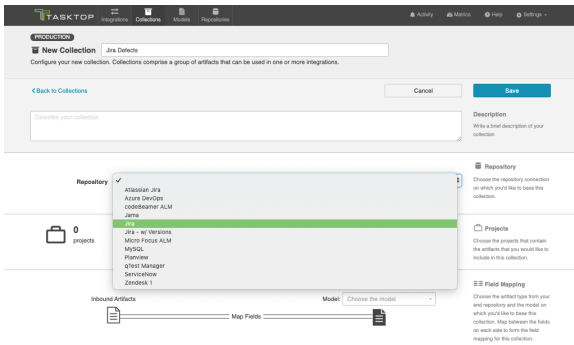
Select **Work Item Collection** as the collection type.



Name and describe your collection.

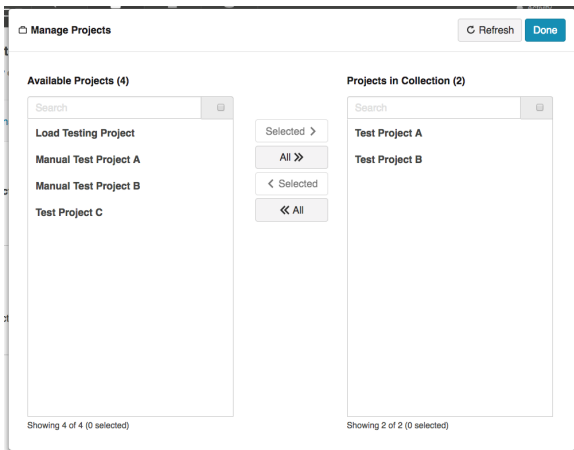


Select the repository that you would like to connect to. The collection will include artifacts from the repository you have selected.

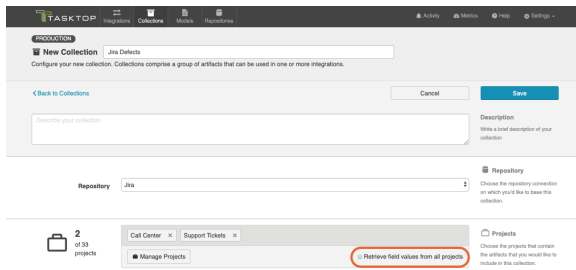


Add projects to your collection by selecting **Manage Projects**. These are the projects from which Tasktop will be able to create, retrieve, and update artifacts.


**Note:** In some cases, the word **Project** is used loosely. You may be selecting workspaces or some other organizational structure, depending on the repository you've connected to. You can review our [Connector Docs](#) to see which containers are supported for each repository.

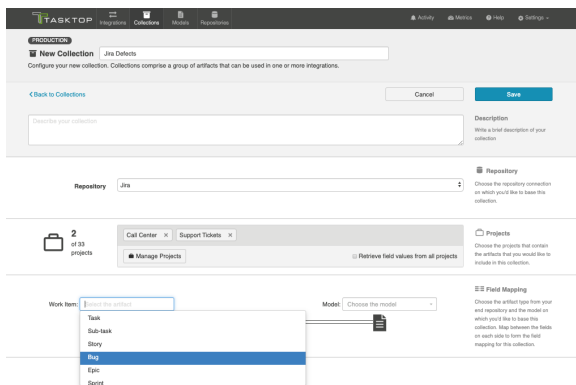


By default, Tasktop retrieves field values from one sample project for mapping. In rare cases where values vary between projects, check the **Retrieve field values from all projects** box on the Collection configuration screen to retrieve all possible values. Be aware that retrieving values from all projects can take some time.

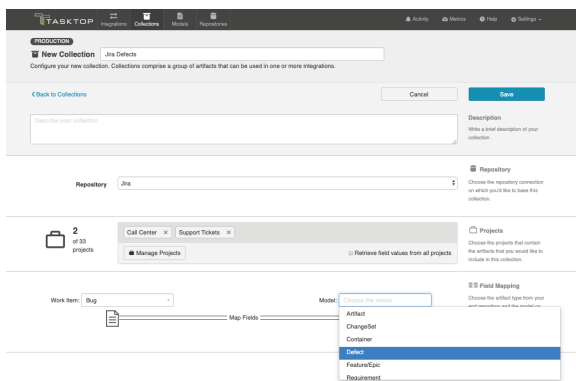



Select the artifact type from the repository that you would like to include in this collection.

 For instructions on how to include multiple artifact types in a single collection, see the [Affinity Modeling](#) section below.



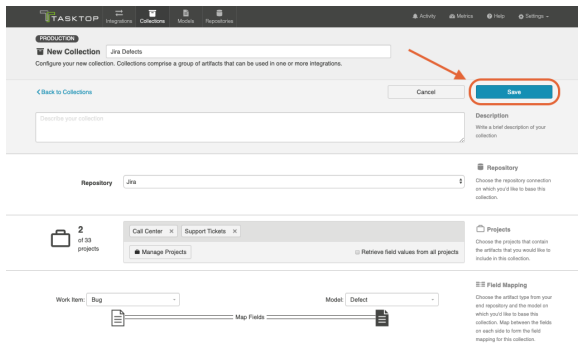
Select the model you'd like to use for this collection.



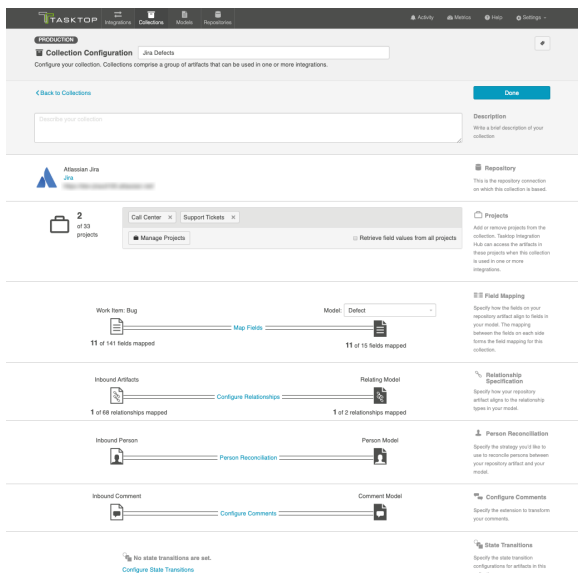
 **Note:** The projects included in your collection must contain at least one artifact of the type selected. For example, in the image above, there must be at least one bug in Test Project A in Jira in order for your collection to save.

Click **Save**.

 **Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your collection.



Once you save, you'll see a number of configuration panels appear:



**Tip:** Each configuration panel is an important part of configuring your collection. Make sure you review the links below to ensure you've configured each section appropriately.

## Map Fields

Clicking **Map Fields** will take you to the Field Mapping screen. On this screen, you will be able to specify how fields in your repository are mapped to fields in your model. This mapping will determine how information flows between fields in your source and target collection.

You can learn more about this process on the [Field Mapping](#) page.

## Map Test Step Fields

Depending on your [Tasktop edition](#), you may see an option to **Map Test Step Fields**.

You can learn more about this process on the [Test Synchronization](#) page.

## Configure Relationships

Clicking **Configure Relationships** will take you to the Relationship Specification screen. On this screen, you will be able to specify how **relationship** fields in your repository are mapped to fields in your model. Relationship fields, such as **blocked by**, **is related to**, and **parent**, enable you to preserve the relationship structure between artifacts as you flow information from one collection to the other.

You can learn more about this process on the [Relationship Specification](#) page.

## Person Reconciliation

Clicking **Person Reconciliation** will take you to the Person Reconciliation screen. On this screen, you will be able to specify the strategy you'd like to use to reconcile person fields between your repositories.

You can learn more about this process on the [Person Reconciliation](#) page.

## Comment Configuration

Clicking **Configure Comments** will take you to the Comment Configuration screen. On this screen, you will be able to apply a comment extension.

You can learn more about this process on the [Comment Configuration](#) page.

## State Transitions

Clicking **Configure State Transitions** will take you to the State Transition screen. On this screen, you will be able to configure state transitions to successfully flow field updates for fields that require defined workflows within your repository.

You can learn more about this process on the [State Transitions](#) page.

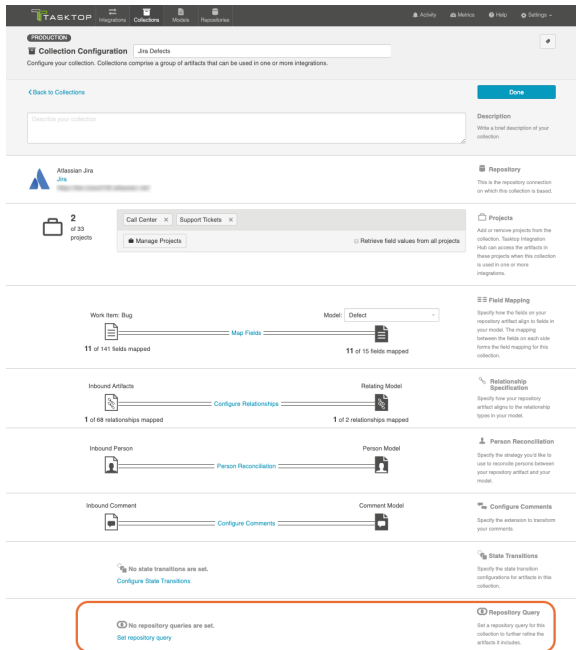
## Artifact Unions

Clicking **Configure Artifact Unions** will take you to the Artifact Unions screen. On this screen, you can configure artifact unions to specify related artifacts whose fields you'd like to flow along with the artifacts in your collection.

You can learn more about this process on the [Artifact Unions](#) page.

## Optional: Set a Repository Query

If you have enabled repository queries for the repository that you have connected to, you will also see a **Repository Query** sash at the bottom of the screen.



**Note:** Repository Queries are advanced functionality, and should only be used when you are truly unable to filter as desired using the built-in Tasktop functionality of Repositories, Collections, and Artifact Filtering.

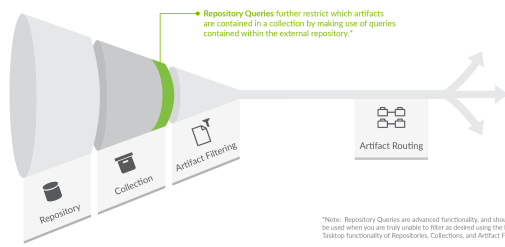
When configuring your integration, you have several options available to refine which artifacts are eligible to flow.

- First, by defining your **repository** (for example, Jira)
- Next, when creating your **collection**, you further refine which artifacts are eligible to flow by selecting only one **artifact type** (for example, defects), and **one or more projects** within your repository.
- Next, by configuring **artifact filtering** at the **integration** level, you further refine which artifacts can flow, based on **fields on those artifacts**,
- And finally, by configuring **artifact routing**, you determine which projects from your collection will participate in the integration, as well as where new artifacts will be created and updated, based on the projects they originated in.

**Note:** When setting a repository query, we recommend only including artifacts of the same type and in the same project(s) as the collection. If a repository query contains many artifacts not included in the collection, performance may be impacted.

In general, the options outlined above should allow you the flexibility to create collections that are broad enough to be reusable in a range of integrations, while still having fine-grained control at the integration-level to ensure that only desired artifacts are flowing within the context of that integration.

In rare cases, however, you may find that the best option to restrict the artifacts eligible to flow is by setting a query within the repository itself.



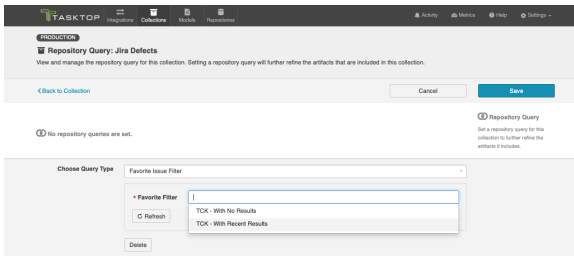
\*Note: Repository Queries are advanced functionality, and should only be used when you are fully unable to filter as desired using the built-in Tasktop functionality of Repositories, Collections, and Artifact Filtering.

If you plan to utilize repository queries, check the box next to **Enable collections to be refined by setting a repository query**, on the **Repository Connection** screen.

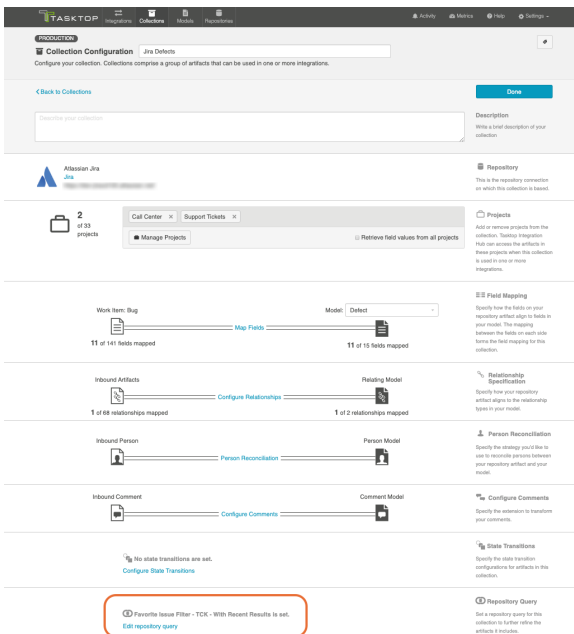
Once this is selected, you will be able to select a repository query at the Collection level for any collections utilizing this repository.

On the Repository Query screen, you can search for your desired repository query. Select the query you'd like to use, and click **Save** and then **Done**.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your repository query.



You will then see the selected repository query on the Collection Configuration screen:



**Note:** Remember, applying a repository query to a collection will only further refine the artifacts included in that collection. If you select a query that encompasses artifacts in projects not in your collection, these artifacts will not be added to the collection unless you also add those projects to your collection as you normally would.

## Affinity Modeling

**Affinity Modeling** enables you to model multiple work item types of similar type in a single collection. This means modeling multiple projects and artifact types simultaneously, with fewer collections and field mappings to configure.

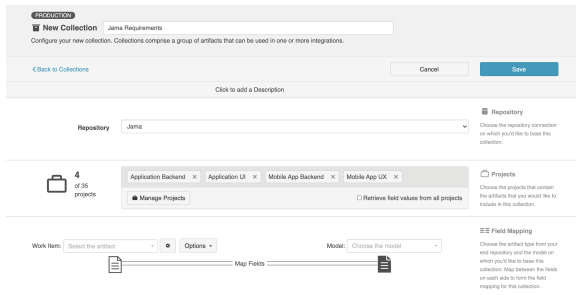
**Warning:** Please be aware of the following limitations before enabling affinity modeling:

- Repositories whose projects depend on type (e.g., Micro Focus PPM) are not eligible
- Repositories that do not allow project selection (e.g., ServiceNow, BMC Remedy) are not eligible
- Collections configured with a container type are not eligible

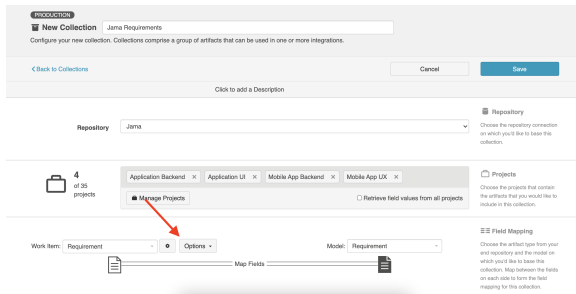


- Collections with affinity modeling enabled cannot be used in Container + Work Item Synchronizations

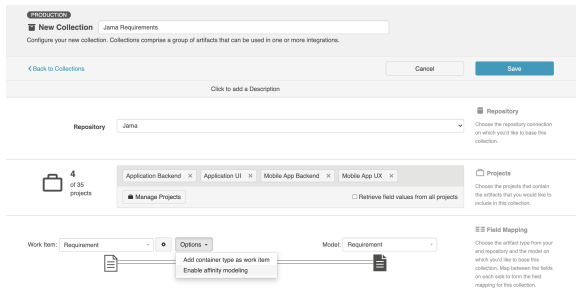
To configure affinity modeling, navigate to the **Collection Configuration** screen.



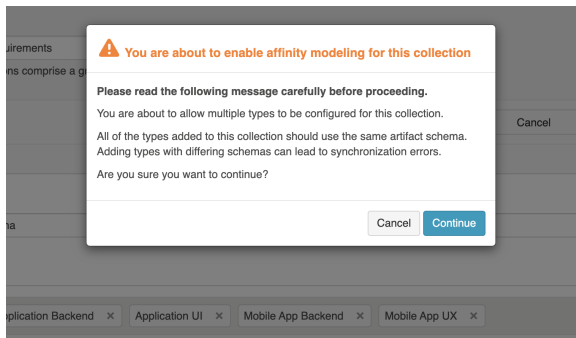
On this screen, you'll see an **Options** dropdown next to the work item type.



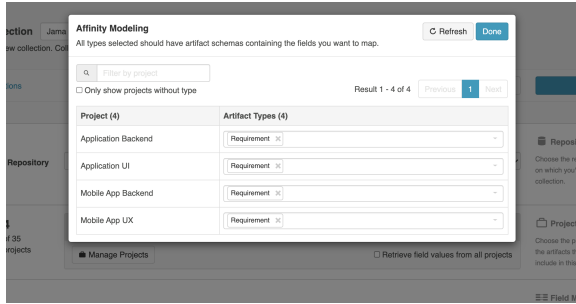
Select **Enable affinity modeling**.



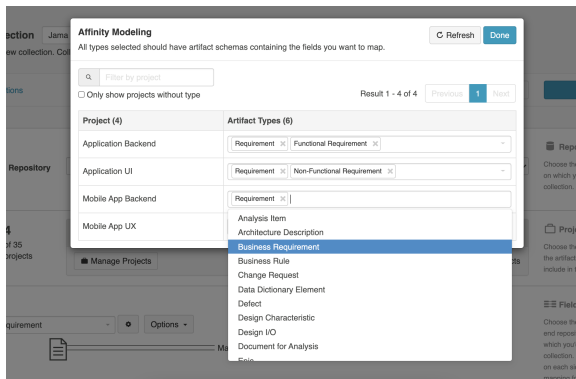
A warning message will appear informing you that the artifact types in the collection should use the same artifact schema to prevent synchronization errors.



After clicking **Continue** on the warning message, the affinity modeling pop-up will appear.

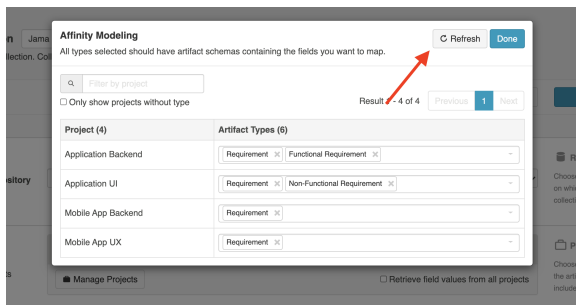


Here, you can add multiple artifact types to the project(s) in your collection.

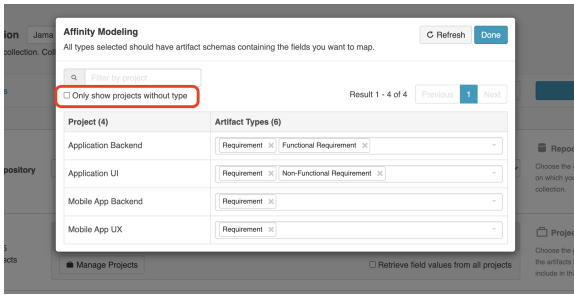


**Note:** All included artifact types should share the same artifact schema to prevent synchronization errors.

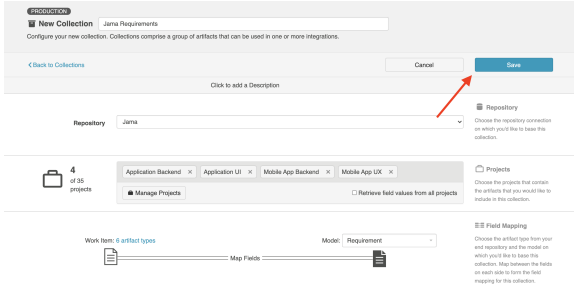
If you need to refresh the fields available for the collection, use the **Refresh Fields** button in the Hub UI, rather than your browser's refresh button.



If you encounter an error when saving the collection due to a project missing an artifact type, enable **Only show projects without type** to identify which projects need to be updated.

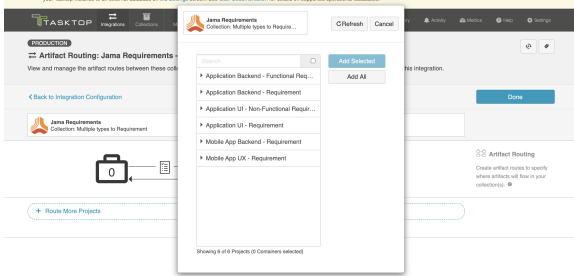


To save your changes, click **Save**.



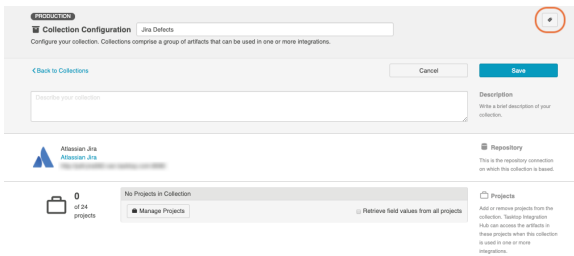
Once configured, you will see all of the types available to be routed on the Artifact Routing screen when configuring your integration.

**Note:** Any existing field mappings will be reused for newly added types.



## Viewing Associated Configuration Elements

To view associated configuration elements (such as models or integrations that utilize the collection you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



 **Associated Elements for Repository Collection "Jira Defects"**

- **1 Model used by this Repository Collection**
  - [Defect](#)
- **1 Repository Connection used by this Repository Collection**
  - [Jira](#)
- ≡ **1 Integration using this Repository Collection**
  - [HP and Jira Synchronizations](#)

Close

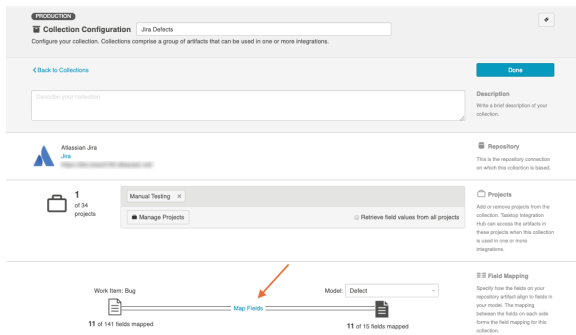
# Field Mapping

## Introduction

After saving your [Work Item Collection \(Repository\)](#), the next step is to map fields from your collection to your model. This will tell Tasktop how to flow information to and from your collection.

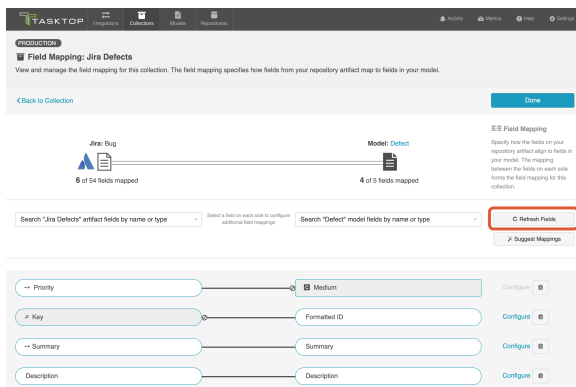
## Mapping Fields

After saving your [Work Item Collection \(Repository\)](#), you'll see that the **Map Fields** link becomes active.

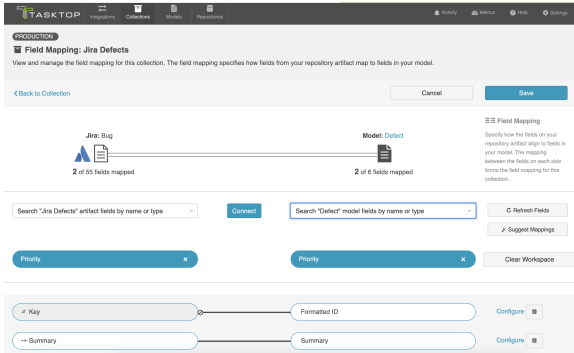


Clicking this link will take you to a drill in page where you can specify how the fields in your model will map to the fields available on the artifact within your repository. Tasktop will auto-map fields when possible based on the names of fields and the smart field designations that have been set in a given model.

**Tip:** If you need to refresh the fields available for the collection, use the **Refresh Fields** button in the Hub UI, rather than your browser's refresh button.



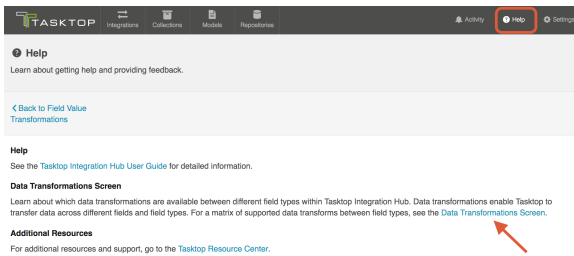
You can map additional fields by using the two drop down boxes:



## Transforms

When you map a collection field to a model field, it is necessary to **transform** the data from the source field to the target field. Depending on the field types, that transform may or may not be possible within Tasktop Integration Hub.

You can see a table of the available transforms by clicking the **Data Transformations Screen** link on the Help page.



This will lead you to the Field Value Transformations screen. Here, you can see which collection-to-model field type transformations are available.

Supported Collection Field Types	Supported Model Field Types															
	Boolean	Date	Date Time	Double	Duration	Location	Long	Multi Select	Person	Persons	Relationship	Relationships	Rich Text	Single Select	String	Web Links
Boolean	●															
Container																
Containers																
Date		●														
Date Time		●	●													
Double				●												
Duration				●	●											
Float				●												
Integer						●										
List of String							●									
Location					●											
Long							●									
Multi Select				●	●			●								
Person								●	●							
Persons								●	●							
Relationship										●	●					
Relationships										●	●					
Rich Text												●	●			
Single Select													●	●		
String		●	●	●	●									●	●	
Web Links															●	●
WebLink																●

You can even filter by Collection to see the specific field labels and field types for that collection:

## Field Value Transformations

See the transformations available between different field types within Tasktop Integration Hub

### Filter table by collection

### Filter table by collection

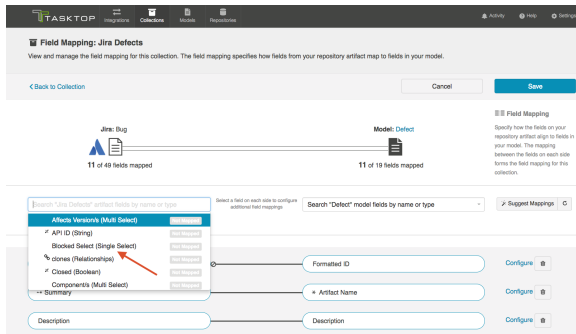
Jira Defects Clear

#### Displaying transformations for field types from repository collection Jira Defects

Supported Collection Field Types	Field Labels	Supported Model Field Types								
		Date Time	Location	Person	Relationship	Relationships	Rich Text	Single Select	String	Web Links
Boolean	<ul style="list-style-type: none"> <li>Closed</li> </ul>							●	●	
Date	<ul style="list-style-type: none"> <li>Due Date</li> <li>Finish Date</li> <li>Start Date</li> </ul>	●							●	
Date Time	<ul style="list-style-type: none"> <li>Created</li> <li>Resolved</li> <li>Updated</li> </ul>	●							●	
Double	<ul style="list-style-type: none"> <li>Fraction Complete</li> </ul>								●	
Duration	<ul style="list-style-type: none"> <li>Original Estimate</li> <li>Remaining Estimate</li> <li>Time Spent</li> </ul>								●	
Location	<ul style="list-style-type: none"> <li>Custom URL</li> <li>url</li> <li>URL</li> </ul>		●				●		●	
Multi Select	<ul style="list-style-type: none"> <li>Affects Version/s</li> <li>Components</li> <li>Custom nFeed field</li> </ul> <a href="#">Show More (7)</a>				●	●		●	●	
Person	<ul style="list-style-type: none"> <li>Assignee</li> <li>Reporter</li> </ul>			●					●	
Persons	<ul style="list-style-type: none"> <li>Watchers</li> </ul>			●					●	
Relationship	<ul style="list-style-type: none"> <li>Epic Link</li> </ul>		●		●	●	●	●	●	●
Relationships	<ul style="list-style-type: none"> <li>blocks</li> <li>clones</li> <li>duplicates</li> </ul> <a href="#">Show More (6)</a>				●	●	●	●	●	●
Rich Text	<ul style="list-style-type: none"> <li>Affects Requirement</li> <li>Affects Test Result</li> <li>Change Set List</li> </ul> <a href="#">Show More (10)</a>						●		●	
Single Select	<ul style="list-style-type: none"> <li>Booleanfake</li> <li>Company</li> <li>Direct Cover Status</li> </ul> <a href="#">Show More (15)</a>				●	●		●	●	
String	<ul style="list-style-type: none"> <li>Alternate URL</li> <li>API ID</li> <li>Design URL</li> </ul> <a href="#">Show More (17)</a>	●	●	●			●	●	●	
Web Links	<ul style="list-style-type: none"> <li>Web Links</li> </ul>									●

On the Field Mapping screen, if you attempt to map fields that do not have a valid transform between one another (for example, if you map 'due date,' a date field, to 'status,' a single-select field), you will get an 'invalid mapping' warning, and the mapping will not be saved.

To help troubleshoot, you can review the field type when selecting each value from the drop down menu. This will enable you to ensure that the transform between the two field types is supported.



## Field Mapping Icons

On the Field Mapping screen, you will see a number of icons which will help you understand any special properties or requirements for each field. If you hover your mouse over an icon, you will see a pop-up explaining what the icon means. You can also review their meanings in the legend below:

Icon	Meaning
	<p>A constant value will be sent. Note that:</p> <ul style="list-style-type: none"> <li>If the icon is on the side of the collection, this means that a constant value will be sent to your model. This means that any time this collection is integrated with another collection, the <i>other</i> collection will receive this constant value for the field in question.</li> <li>If the icon is on the side of the model, this means a constant value will be sent to your collection. This means that any time this collection is integrated with another collection, that <i>this</i> collection will receive the constant value for the field in question.</li> </ul>
	<p>A state transition will be utilized. Note that:</p> <ul style="list-style-type: none"> <li>If the icon is on the side of the collection, this means that a <a href="#">state transition graph</a> is being utilized.</li> <li>If the icon is on the side of the model, this means that a <a href="#">state transition extension</a> is being utilized.</li> </ul>
	Repository field is read-only and cannot receive data.
	To create artifacts in your repository, this field must be mapped to your model.
	This is a required field in your model; it must be mapped to your collection.
	This field will not be updated as part of your integration because the mapping would be invalid. You do not have the option of changing this.



# Constant Value Mapping

In some scenarios, either the collection artifact or the model might require that a value be provided for a given field. This value is usually provided by mapping it to the equivalent field in the collection or model. However, sometimes your collection artifact has a field that needs a value that doesn't align with any fields in your model, and sometimes your model might have a required field that doesn't have an equivalent field from the collection artifact. In these cases, you can set a constant value. By doing so, you'll specify the value that you would like to provide for that field.

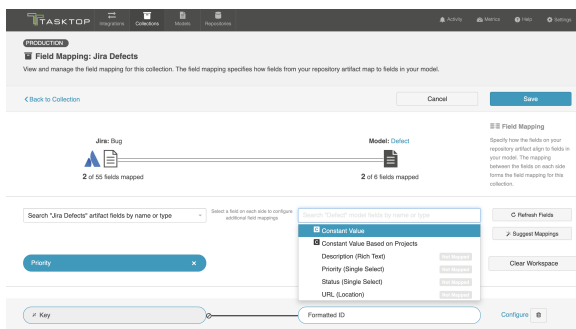
Constant values can be set for the following field types:

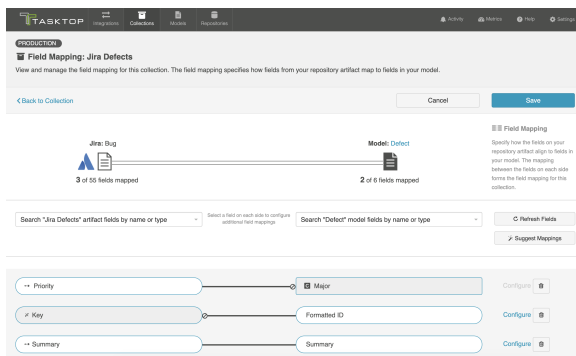
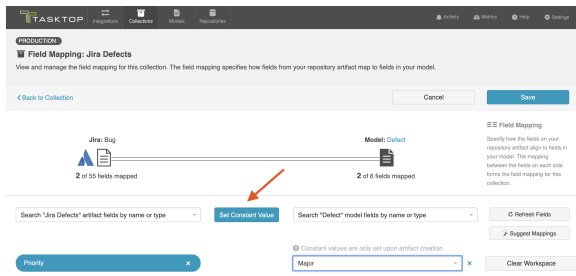
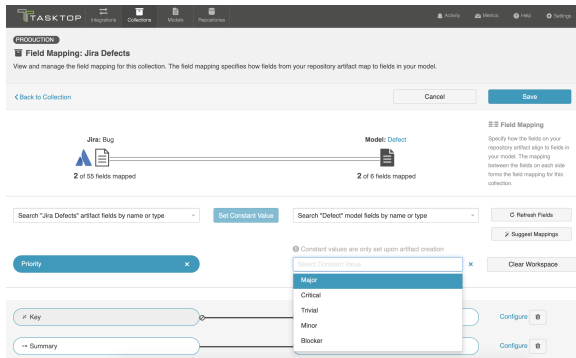
- Boolean
- Date/DateTime
- Double
- Location
- Long
- Multi-Select
- Person
- Rich Text
- Single-Select
- String

## Scenario 1: If your repository requires a field for artifact creation, but that field is *not* a part of your model:

Solution: Set a constant value on the side of the model, to send to your collection.

To set a constant value for a field, select **Constant Value** from the drop down menu on the model side. Enter the value, and then click the **Set Constant Value** box.





Once the constant value is set, you will notice a few things:

- The pill will be rectangular and grey: this denotes that a constant value has been set.
- The Constant Value icon will be displayed inside the pill.
- The 'prohibited' icon will appear next to the pill. This indicates that no values can be sent to the Constant Value field. The constant value is essentially a dead end, and cannot be linked to a repository or model on the other side.

In the scenario above, any time a new defect is created in Jira, the priority will be set to 'Major.' Jira will not send 'priority' data to any other collections, as 'priority' is not mapped to the model.

## Constant Values per Project

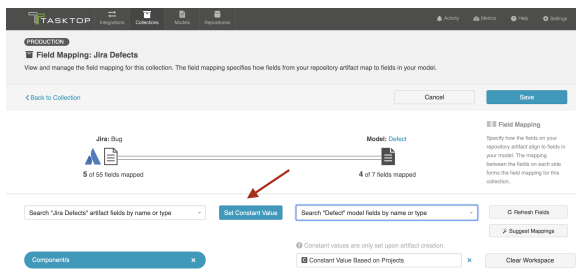
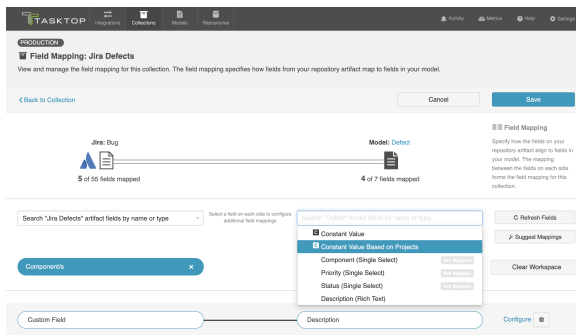
If desired, you can also set constant values per project.

You may wish to set a constant value based on project in the following scenarios:

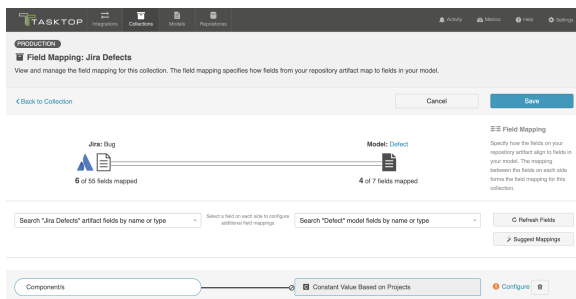
- In order to set a unique value for a specific field, such as release or iteration, depending on the project

- If the values for a single-select field vary across projects

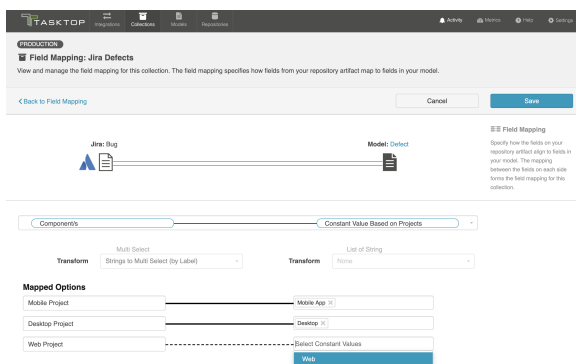
To do this, select **Constant Value Based on Projects**:



Once selected, you will see an orange exclamation point appear next to the 'Configure' link:

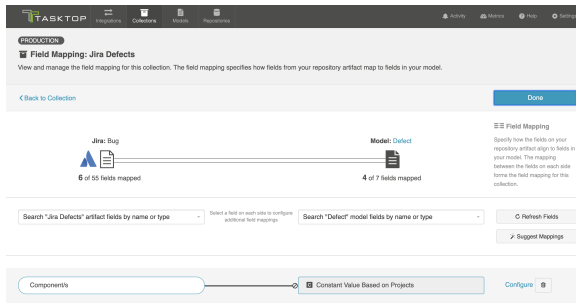


Click **Configure** to get to the Field Configuration Screen. On this screen, you will be able to set a distinct constant value for each project in your collection:



In the screenshot above, a bug created in the Desktop Project would have the value **Desktop** applied to the Component(s) field, while a bug created in the **Mobile Project** would have the value **Mobile App** applied to the Component(s) field.

lied to the Component(s) field, and finally a bug created in the Web project would have the value **Web** applied to the Component(s) field.



Once the constant value is set, you will notice a few things:

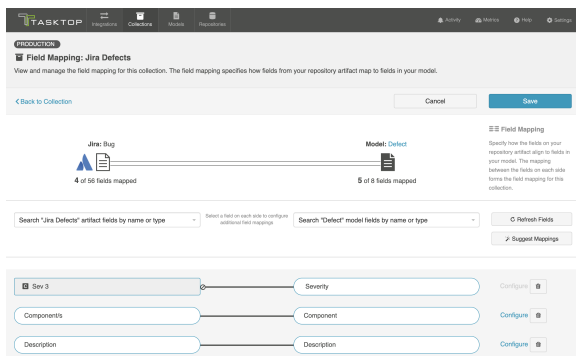
- The pill will be rectangular and grey: this denotes that a constant value has been set.
- The Constant Value icon will be displayed inside the pill.
- The 'prohibited' icon will appear next to the pill. This indicates that no values can be sent to the Constant Value field. The constant value is essentially a dead end, and cannot be linked to a repository or model on the other side.

**Note:** Sometimes, a single-select field in your collection will not return any values to be selected in in the UI. In cases when this is true, and when the artifact will accept new values for that field, you will see a text input in which you can configure a constant string value (instead of the traditional drop-down list for a single-select).

## Scenario 2: If your model requires a field, but the repository utilized in your collection does not have that field:

Solution: Set a constant value on the collection side to send to your model. This means that any time your source collection creates a corresponding artifact in a target collection, the field will automatically be set to the constant value in the target repository.

To set a constant value for a field, select **Constant Value** from the drop down menu on the collection side. Enter the value, and then click the **Set Constant Value** box.



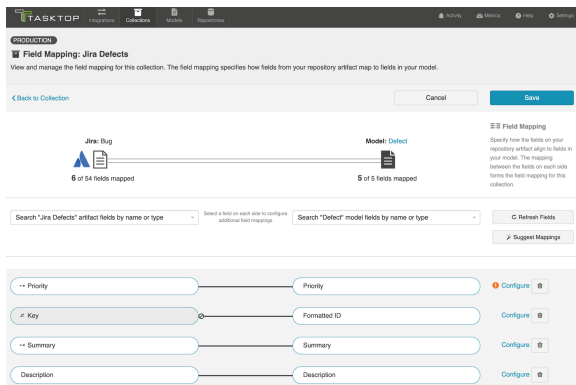
Once the constant value is set, you will notice a couple of things:

- The pill will be rectangular and grey: this denotes that a constant value has been set.
- The Constant Value icon will be displayed inside the pill.
- The 'prohibited' icon will appear next to the pill. This indicates that no values can be sent to the Constant Value field. This makes sense, because in this example your repository did not have a 'severity' field to begin with.

In the example above, any defects that flow from Jira to a target repository will populate the 'Severity' field in the target repository with a value of 'Sev 3.'

## Field Configuration

Once your collection-to-model field mapping is complete, your next step is to configure each field. Tasktop will generally auto-configure these for you, but in certain cases (such as single-selects and multi-selects), additional configuration may be needed. In scenarios where the integration cannot run successfully without additional configuration, you will see an orange configuration warning next to that field.

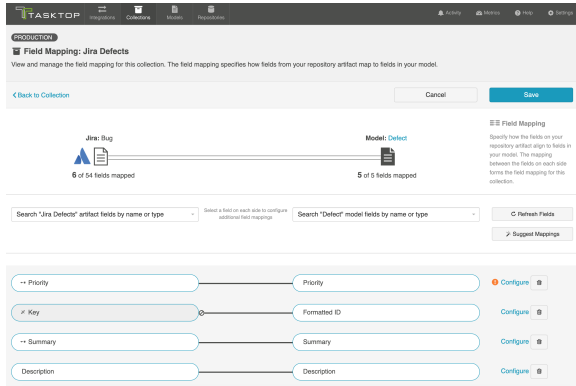


To review and update an individual field's configuration, click the **Configure** link to its right. You can learn more about Field Configuration on the [Field Configuration](#) page of our User Guide.

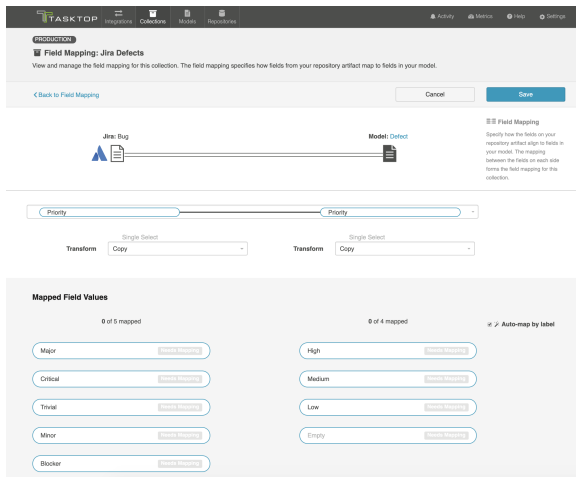
# Field Configuration

## Introduction

Once your [collection-to-model field mapping](#) is complete, your next step is to configure each field. Tasktop will generally auto-configure these for you, but in certain cases (such as single-selects and multi-selects), additional configuration may be needed. In scenarios where the integration cannot run successfully without additional configuration, you will see an orange configuration warning next to that field.



To review and update an individual field's configuration, click the **Configure** link to its right. This will lead you to the **Field Configuration** screen:

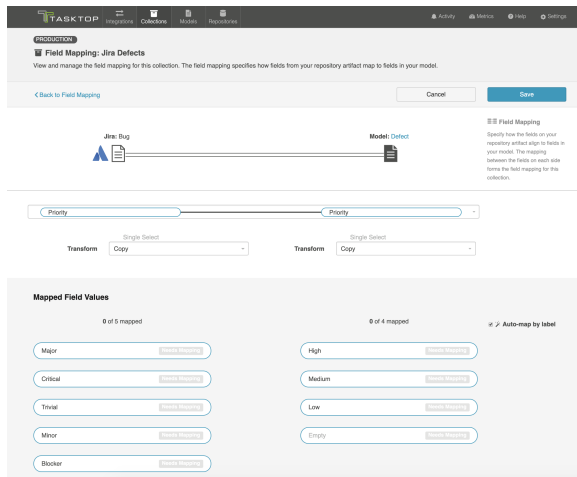


## Transforms

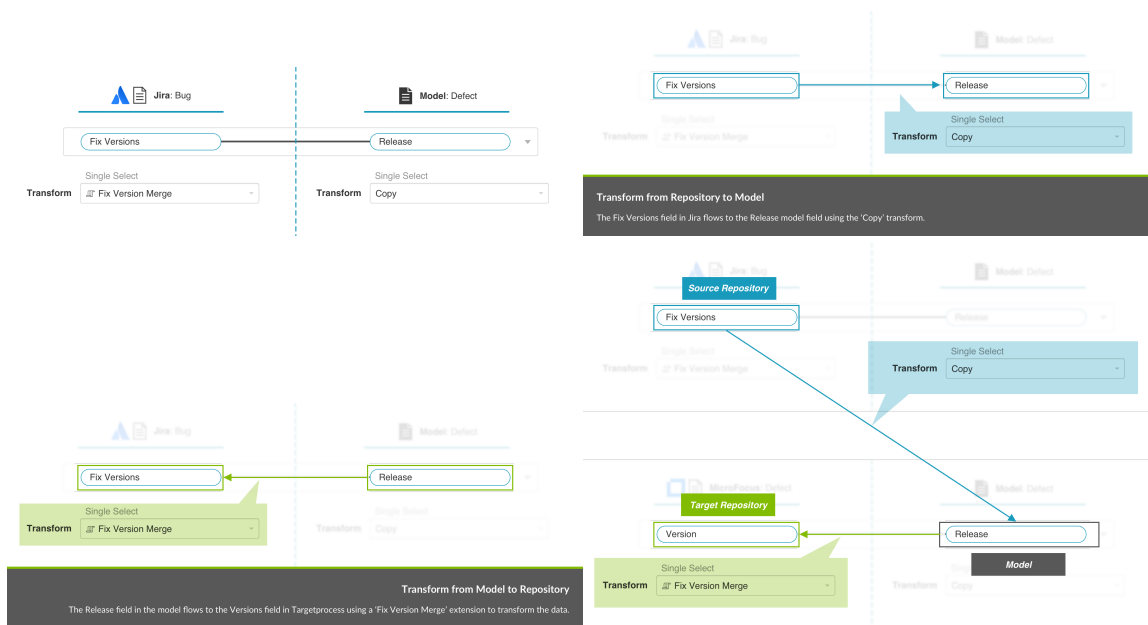
The Field Configuration screen is where you can configure your transforms and value mappings.

Similar fields in different repositories often come in different formats, resulting in the need for values to be transformed to their proper format before flowing to the target repository. This screen allows you to configure how different types of fields will translate from one repository to the other.

You can learn more about Supported Transforms on the [Field Mapping](#) page.



The transform on the left will impact how data flows into your repository (from your model), and the transform on the right will impact how data flows into your model (from your repository).

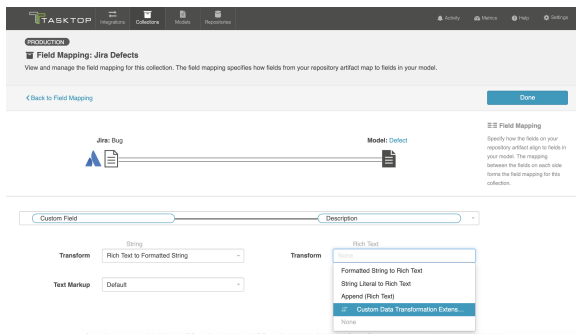


Here are some examples of available transforms:

- **Copy:** A copy of the value from the source field will flow to this field. The value sent will overwrite whatever was previously held in that field.
- **Append:** A copy of the value from the source field will flow to this field. Values that existed previously will remain, with the new value appended to the end. This transform is typically utilized within the context of a [Modify via Gateway Integration](#).
- **None:** No value will flow from the source field to this field.
- **(Field Type) to (Field Type), for example 'Formatted String to Rich Text':** In some cases, you may need to transform the data from one field type (such as a Formatted String) to another field type

(such as Rich Text). In this scenario, your transform will function similarly to the 'copy' transform. It will overwrite whatever values were previously held in that field with the new (transformed) value sent from the source field.

- Note that for transforms for multi- field types (i.e., multi-select, containers, relationships, etc), where appropriate, the values will be listed out and separated by a comma. For example, a "Containers to ID" transform will flow all container IDs, each separated by a comma, to a string field.
- **Rich Text to Literal String** and **Literal String to Rich Text**: While 'rich text to formatted string' strips the text of any code and outputs a human-readable string, 'rich text to literal string' outputs raw rich text data, preserving the rich text markup (for example, flowing '<b>bold</b>,' rather than 'bold'.
- **Location to Web Link (Summary as Label)**: Some transforms allow Tasktop to perform some behind-the-scenes magic. For example, the 'Location to Web Link (Summary as Label)' transform will flow a location (i.e., the URL for an artifact) to a Web Link field, using the Summary field on that source artifact as the label for that hyperlink.
- **Custom Data Transformations**: If you have configured a [Custom Data Transformation extension](#), you can apply it on this screen:



In most scenarios, the default setting will be appropriate, and you will not need to modify anything here.

## Rich Text (Text Markup)

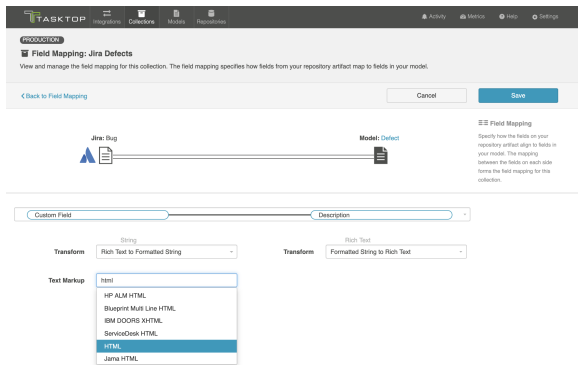
In order to ensure that rich text fields are formatted properly between repositories, the text markup language must be set appropriately. The good news is that Tasktop's default text markup configuration should cover most rich text scenarios.

You'll notice that the Text Markup field is automatically set to 'default.' You can leave this as-is for the majority of integration scenarios.

However, there may be cases where you'd like to customize the text markup language used. For example, you could be using a plug-in like JEditor - Rich Text Editor for Jira, which causes your repository to utilize an unexpected rich text markup language.

In cases like this, you can customize the desired text markup language. You'll see a wide range of text markup options available on this screen. Search for the one you need and select it here. The language selected will impact both how data flows *into* and *out of* the collection for that specific field.

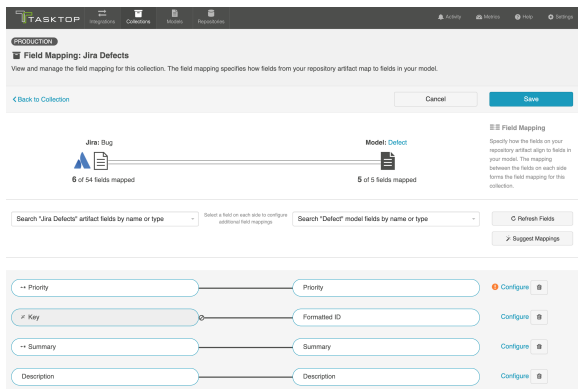




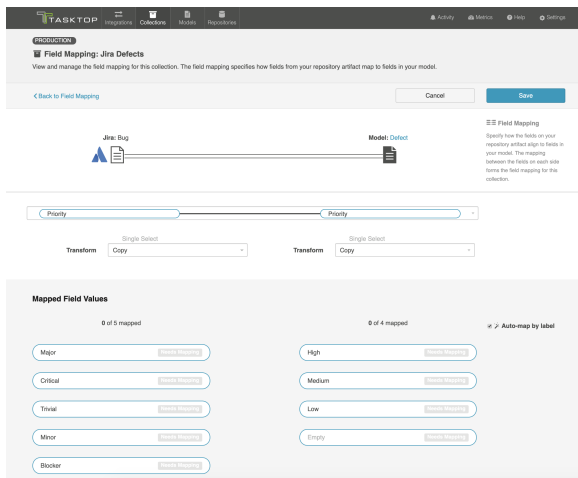
## Single- and Multi-Select Fields

When flowing single- and multi-select fields, Tasktop will need to know how to translate different field values between repositories. To handle this, Tasktop offers an easy-to-use field value mapping canvas on the Field Configuration screen.

If your field values have not yet been mapped, you'll notice an alert next to the **Configure** link on the Field Mapping screen:



Once you click **Configure**, you will be lead to the Field Configuration screen. Please review the sections below in order to learn which selections to make on this screen.



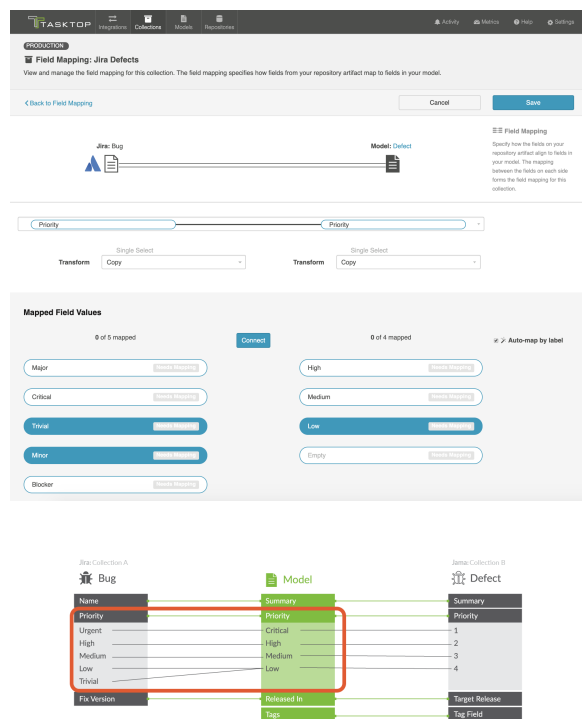
# Transforms for Single- and Multi-Selects

In most single- and multi-select field scenarios, you will configure your transform as 'copy' on both the collection and on the model side. This means that the model will pass an identical copy of its value to the collection, and vice versa. This should be the default setting.

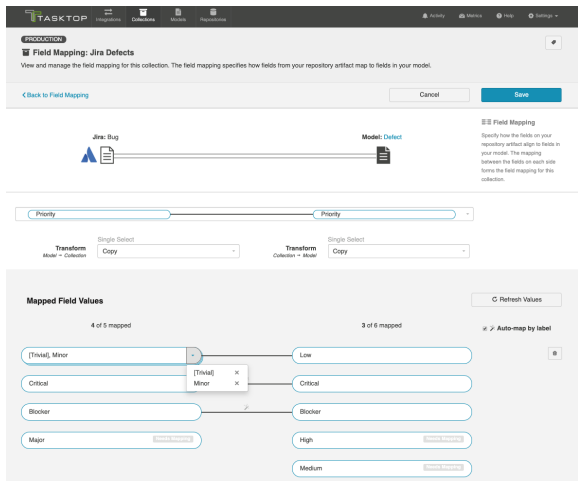
## Field Value Mapping

If the **Auto-map by label** box is checked, Tasktop will use its built-in smarts to pre-map some of the field values for you based on their labels. If you'd like, you can click the trash can icon next to each mapping to remove the mapping map, and then manually re-map them.

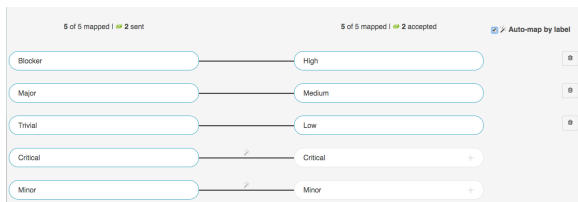
To complete the field value mapping, select the values in the collection and in the model that you would like to map to one another, and then click **Connect**. This process enables to the model to act as a 'translator' between two different collections which may have different sets of values for a single- or multi-select field.



💡 When you map multiple collection values to a single model value, you will find that one value on the collection side is listed in brackets. This indicates which value will be written when the mapped model value is flowed to that field. In the scenario below, if the model passes a 'low' priority value to your collection, that artifact will default to a priority status of 'minor,' rather than 'trivial.' You can modify the default value by clicking the arrow icon on the collection field pill.

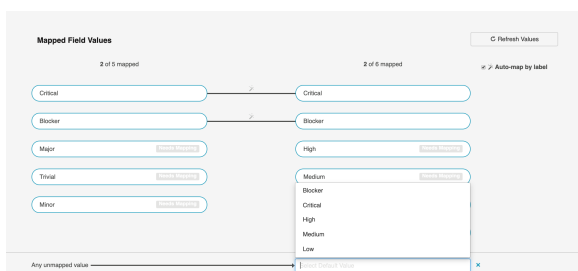


**Note:** If your **model** allows unmapped values to flow for the field you are configuring, you will see an indication of both the number of values that are explicitly mapped to your model, and the number of values that have been 'accepted' by your model. The values that have been 'accepted' are those unmapped values which have been allowed to flow as part of your integration. Note that in most scenarios, the recommended setting is **not** to allow unmapped values to flow. However, allowing unmapped values to flow can make sense in a few specific scenarios, such as an Enterprise Data Stream integration or in single select to string transforms, where there are many options available and you don't desire any normalization of the data flowing through.



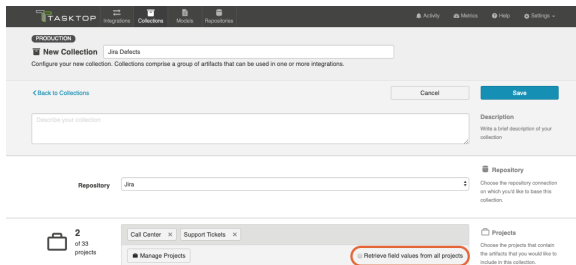
## Default Field Value Mapping

You can set a default field value for any unmapped value by going to the bottom of the **Field Mapping** screen and clicking **Select Default Value**. From the dropdown menu, you can select your default value, and any unmapped value will be set to this value.



## If Field Values Vary by Project

By default, Tasktop retrieves field values from one sample project for mapping. In rare cases where values vary between projects, check the **Retrieve field values from all projects** box on the Collection configuration screen to retrieve all possible values. Be aware that retrieving values from all projects can take some time.



## Specific Use Cases

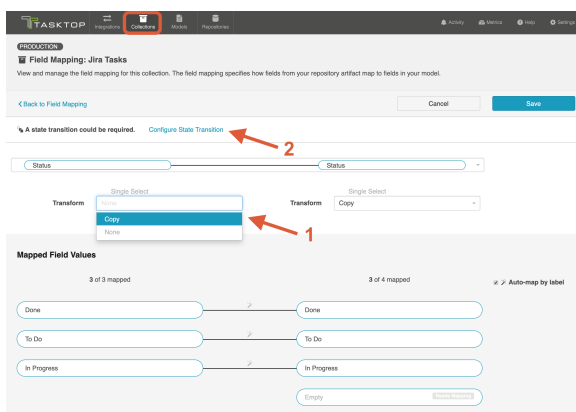
There are a few specific scenarios that will require additional configuration

### State Transitions

Some repositories require that a state transition be performed in order to update the value of certain fields (for example, when an artifact must move from a status of *New* to *In Progress* to *Closed*, but cannot move directly from *New* to *Closed*). If this is the case, you'll notice that the transform on the left for this field defaults to **None**. That is because Tasktop is unable to update that field, unless a state transition has been configured in Tasktop.

If you'd like to configure state transitions for that field, make sure that the field is mapped to the model, and then manually update the transform on the repository side (on the left) to **Copy**. Once the transform is updated, you'll see that the **Configure State Transition** link appears.

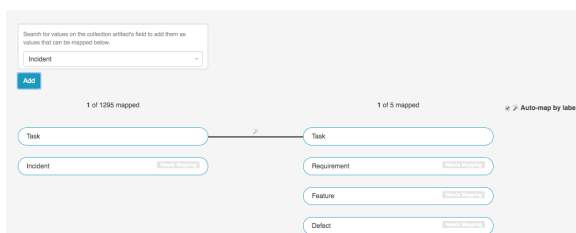
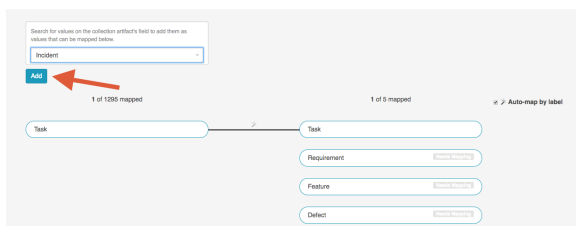
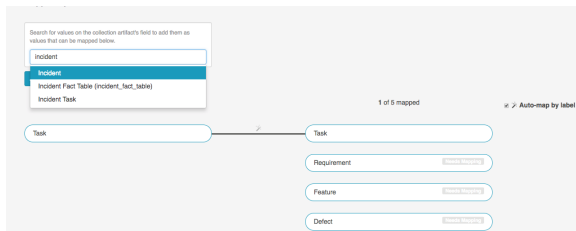
You can learn more about how to configure the state transition on the [State Transitions](#) page.



### Single- or Multi-Select Fields with 25+ Possible Values

If you are mapping a single- or multi-select field that contains over 25 values, you will notice that a search box appears. This is to aid in performance and usability of the Field Configuration screen when mapping a large number of values.

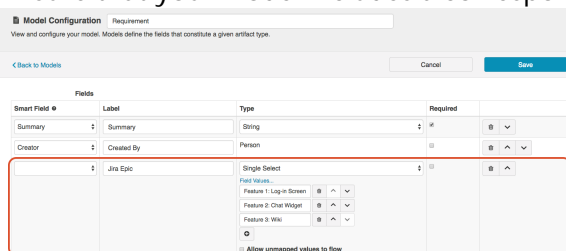
Simply search for the field value you would like to map, and then click **Add**. This will add it to the mapping canvas, so that you can map those values as you normally would.



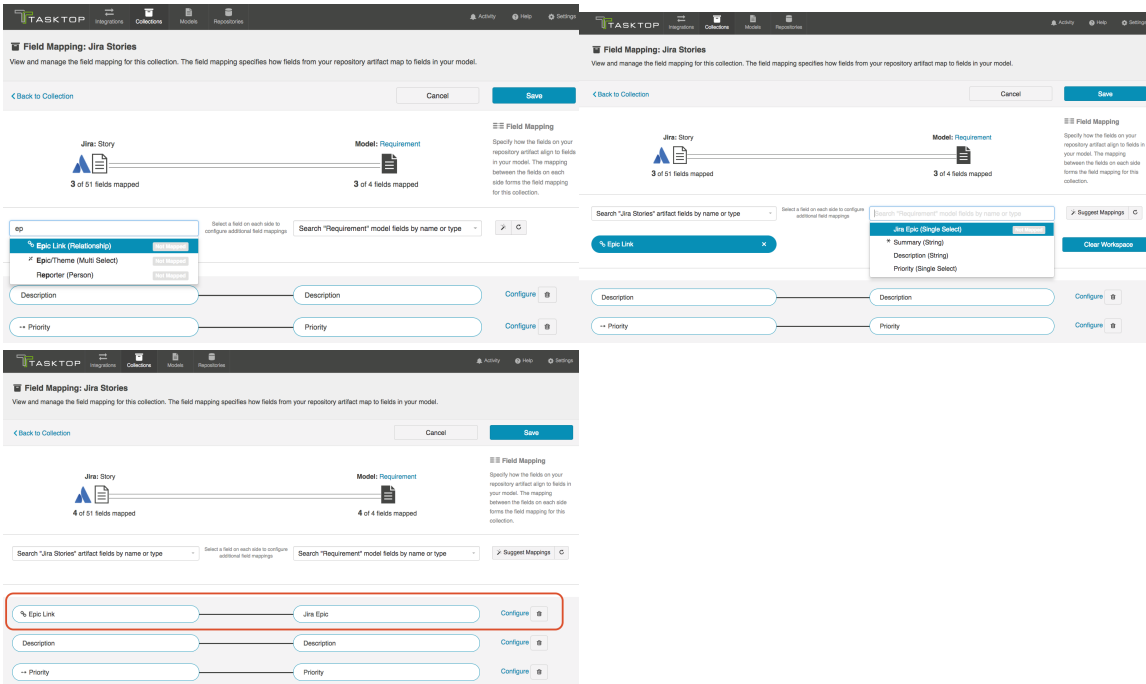
## Relationship to Single-Select Transform

If desired, you can map a relationship on your source artifact to a single-select field on your target artifact. For example, you may wish to write the Jira Epic-link (relationship) to a custom single-select field in qTest Manager. To do this, you must map a relationship field in your source collection to a single-select field in your model.

1. Ensure that your model includes a corresponding single-select field for the mapping

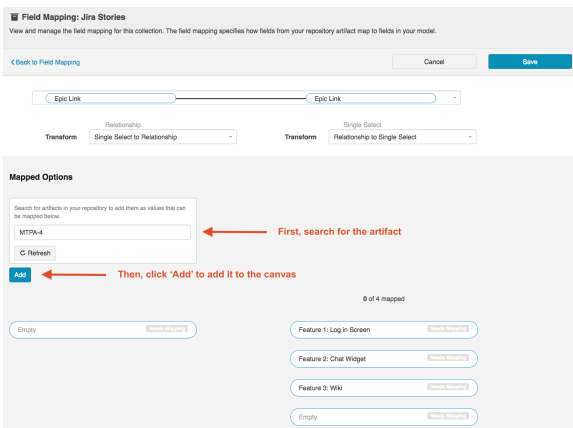


2. In the source collection, click **Map Fields** and create a mapping from the collection's relationship field (Epic-Link in this example) to your model's single-select field.

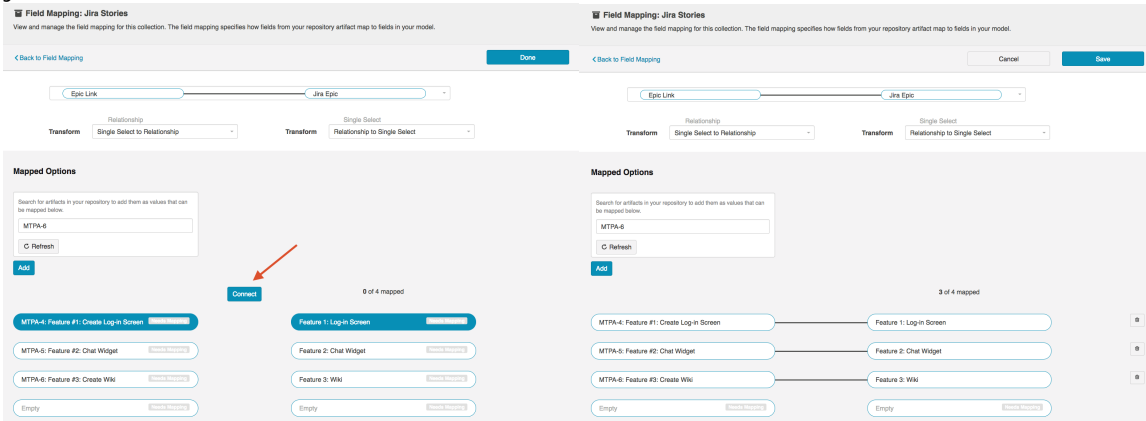


3. Once the fields are mapped, click the **Configure** link on the right side.
4. Here you can search for the related Epics by their **formatted ID**, and click **Add** to add them to your canvas.

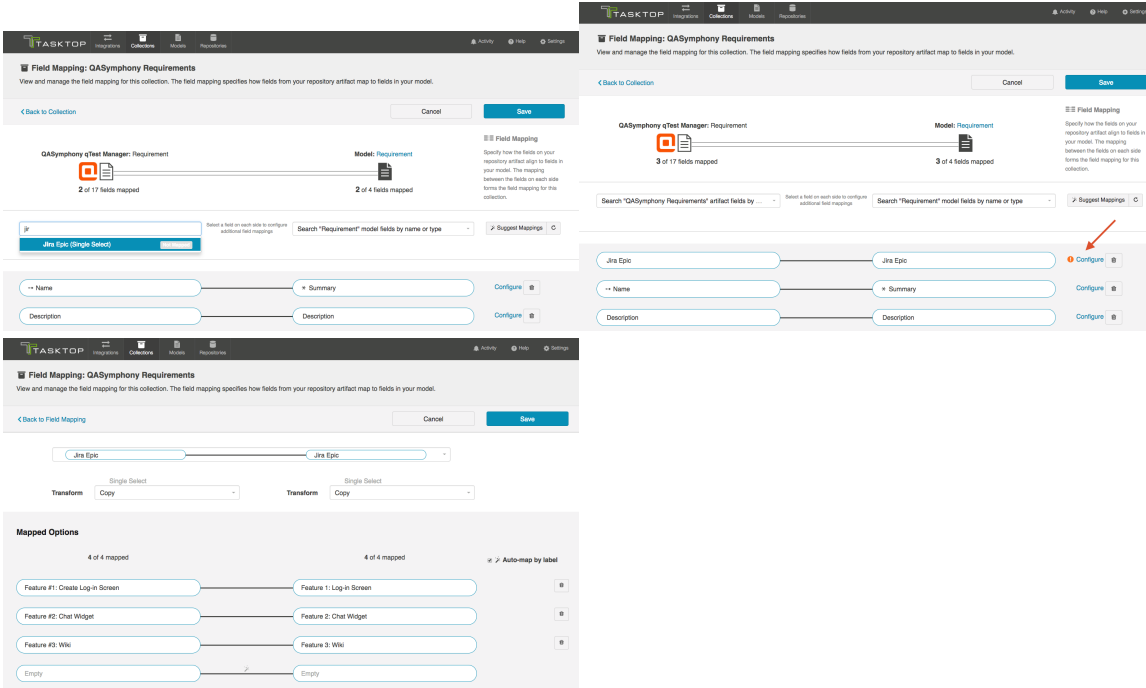
**Note:** if the artifact you are looking for has recently been created in your repository, click the **Refresh** button to refresh the artifacts that Tasktop is aware of. This will allow Tasktop to find that artifact.



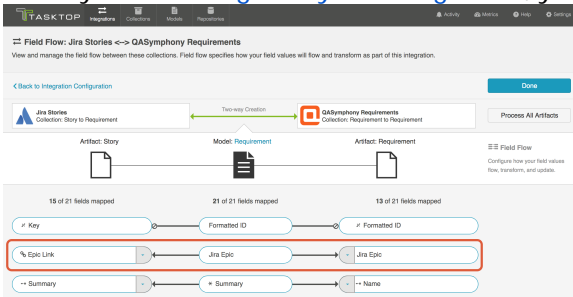
- Once the related Epics are added to the canvas, map them to the available single-select fields in your model.



- Click **Save** and **Done**.
- Navigate to your target collection
- Map the target collection field to the single-select field in your model. Click configure to map the field values.

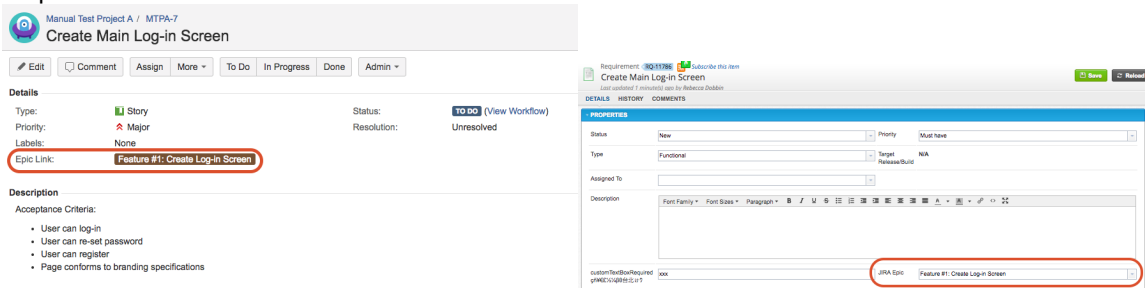


- Once you've **configured your integration**, your completed Integration Field Flow will look like this:



- When you run your integration, the single-select in your target repository will be updated based on the epic link (relationship) in your source repository.

11. Here's the original user story in Jira. You can see that its Epic Link (a relationship to an associated Epic artifact) has flowed to the **Jira Epic** field (a single-select field) on the qTest Manager requirement:



## Next Steps

Once you have completed your [Field Mapping](#) and Field Configuration, your next step will be to review your collection's [Relationship Specification](#) .



# Relationship Specification

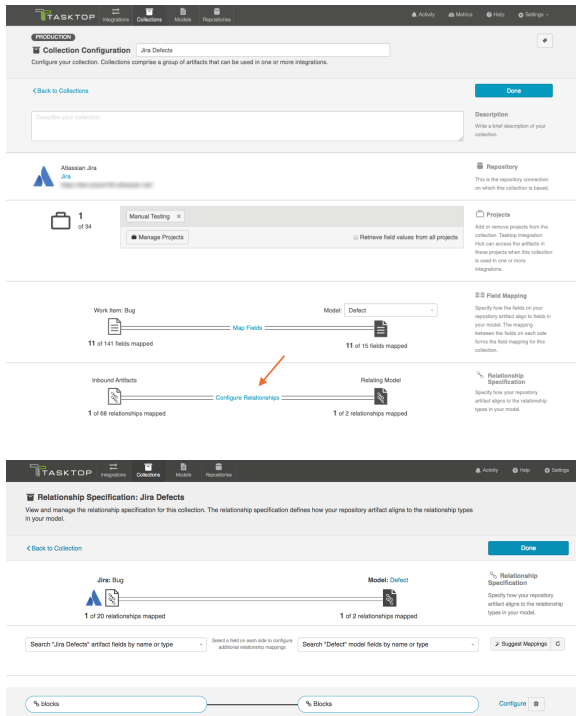
## Introduction

Once you've completed your [Field Mapping](#) and [Field Configuration](#), your next step is to configure your Relationship Specification. The Relationship Specification screen will allow you to specify how **relationship** fields in your repository are mapped to fields in your model. Relationship fields, such as 'blocked by,' 'is related to,' and 'parent,' enable you to preserve the relationship structure between artifacts as you flow information from one collection to the other.

## Configuring Relationship Specification

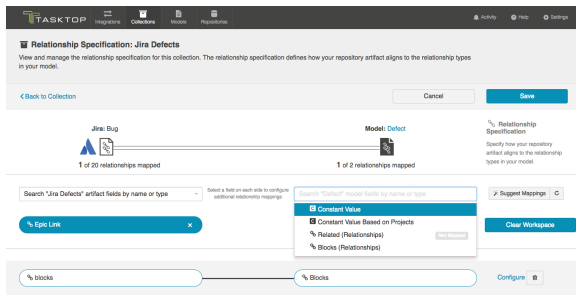
If you have any relationship(s) fields in your model, you can map those to your collection by clicking the **Configure Relationships** link on the **Collection Configuration** screen.

💡 Note that any relationship(s) types you'd like to flow as part of your integration must be mapped to **each** collection involved in the integration.



## Constant Values

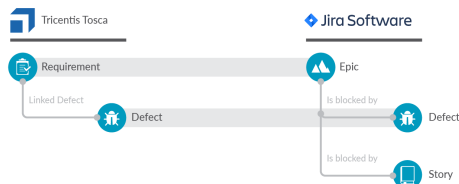
For 'relationship' type fields, you also have the option of configuring constant values. To learn more about constant values, please reference the [constant value section](#) of the Field Mapping page.



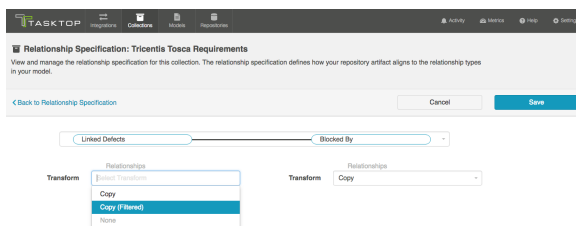
You can learn more about Artifact Relationship Management (ARM) [here](#).

## Filtered Transform

Consider this example scenario: You've mapped the Tricentis Tosca 'linked defect' relationship type to the Jira 'is blocked by' relationship type. In Tosca, the 'linked defect' relationship type can *only* link artifacts to defects. In contrast, Jira's 'is blocked by' relationship type can link artifacts to many different artifact types, such as defects, stories, or epics.



Using the Copy (Filtered) transform on the Tosca side will proactively validate the relationships so that only relationships that will be accepted by the target repository will flow. This can reduce errors in scenarios such as the one described above.



## Additional Information

You can learn more about configuring Artifact Relationship Management (ARM) within the context of a synchronization integration [here](#):

- [Synchronizing Relationships](#)

## Next Steps

Once you have completed your Relationship Specification configuration, your next step will be to review your collection's [Person Reconciliation](#) strategy.

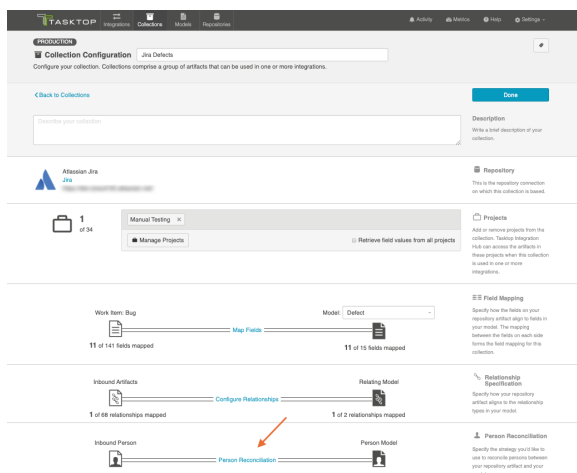
# Person Reconciliation

## Introduction

Once you have completed your [Relationship Specification](#) configuration, your next step will be to review your collection's Person Reconciliation strategy. On this screen, you will be able to specify the strategy you'd like to use to reconcile person fields between your repositories.

## Configuring Person Reconciliation

To configure Person Reconciliation, click the **Person Reconciliation** link.



Tasktop comes with a default person reconciliation strategy ("Copy with Default Matching"), which matches based on name, ID, and/or e-mail.

More specifically, the algorithm will compare the metadata from each side as follows:

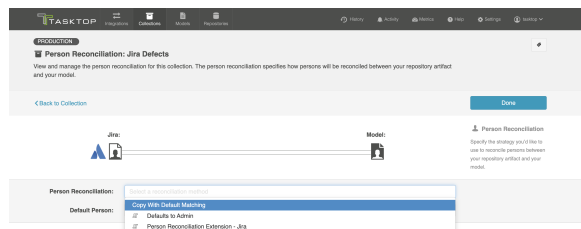
- Username from source to username on target
- Username from source to ID on target
- ID from source to username on target
- ID from source to ID on target
- Email from source to email from target

Please review the [Connector Docs](#) to determine which fields are available for your specific repository. If a field is not available, Tasktop will simply skip that step.

If the default strategy does not cover your needs, you can also configure a [Person Reconciliation extension](#) on the Extensions (Settings) screen, and select that extension here.

We recommend reviewing our [Connector Docs](#) to see each specific connector's unique fields available for Person Reconciliation so that you can better understand your specific use case.

Remember that person fields will flow between your repositories based on the [field flow configuration](#) you've enabled (i.e., one-way, two-way, no update, etc). For a person field not to flow, it must either not be mapped to your collection(s), or be set to 'no update' on the Field Flow screen.

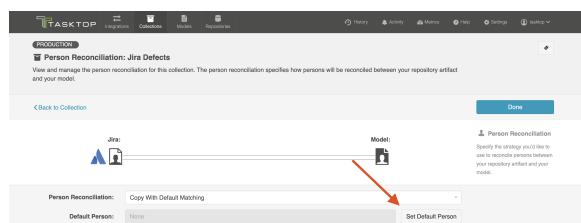


To prevent person not found errors during synchronization, you can set a default person to be used as an override.

**Note:** The default person should only be used to prevent person not found errors. If an extension error appears, the default person should **not** be used.

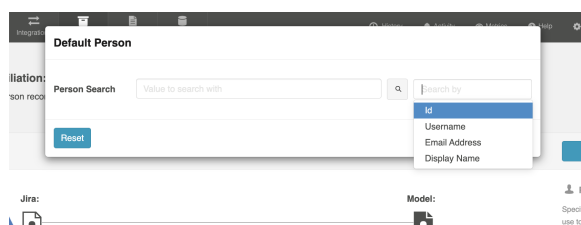
To do this, click **Set Default Person**.

**Note:** The default person reconciliation strategy ("Copy with Default Matching") should be selected.



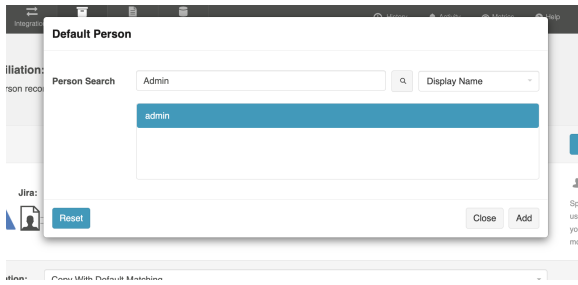
A pop-up box will appear where you can select the field to search by (fields vary based on connector).

**Note:** If you do not select a field to search by, an error will appear.

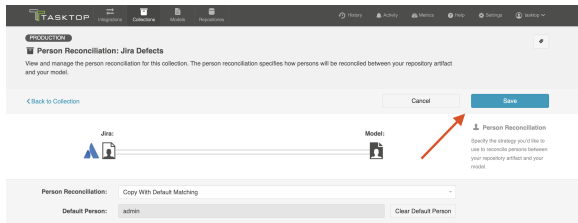


After selecting a field, enter the search value and select the default person.

**Note:** The value entered must be an exact match.



Once you've added the default person, click **Save**.



## Next Steps

Once Person Reconciliation is complete, your next step will be to configure [Comment Configuration](#) if using a comment extension, or [State Transitions](#), if your repository utilizes state transitions or workflows. If not, your collection configuration is complete, and you can move on to [Step 4: Configure your Integration](#).

# Comment Configuration

## Introduction

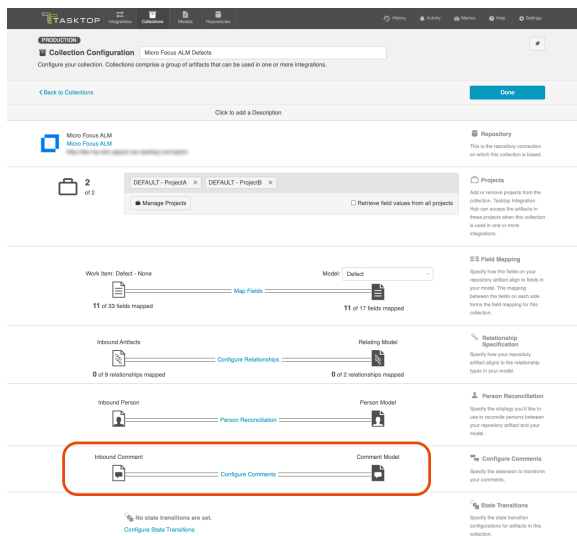
Once you have completed your [Person Reconciliation](#) configuration, your next step will be to review your collection's Comment Configuration. On this screen, you will be able to apply an extension to comment handling for your collection.

## Configuring Comments

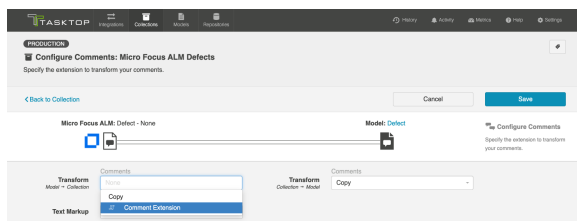
First, create and save your comment extension on the [Extensions \(Settings\)](#) screen. It will be a **custom data transformation** extension.

💡 You can learn best practices for configuring a comment extension [here](#).

Next, navigate to the **Comment Configuration** screen from the **Collection Configuration** screen.

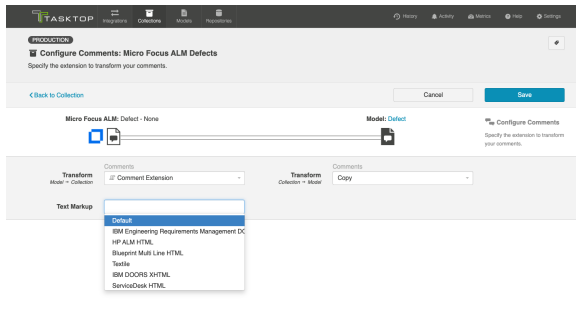


Select the comment extension on the desired side (either impacting data flowing from model collection or collection model):



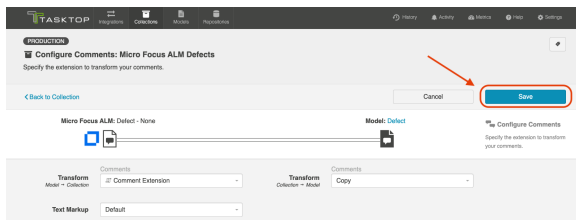
You can also select the text markup language to preserve rich text when flowing comments into and out of your collection.

**Note:** This field is automatically set to default. Tasktop's default text markup configuration should cover most rich text scenarios.



After you have configured your collection's comment settings, click **Save** and **Done** to save your changes.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.



You can enable comment flow for your integration on the [Comment Flow](#) screen.

## Next Steps

Once Comment Configuration is complete, your next step will be to configure [State Transitions](#), if your repository utilizes state transitions or workflows. If not, your collection configuration is complete, and you can move on to [Step 4: Configure your Integration](#).



# State Transitions

## Introduction

Once you've configured your [Person Reconciliation](#) strategy, your next step will be to configure State Transitions, if your repository utilizes state transitions or workflows.

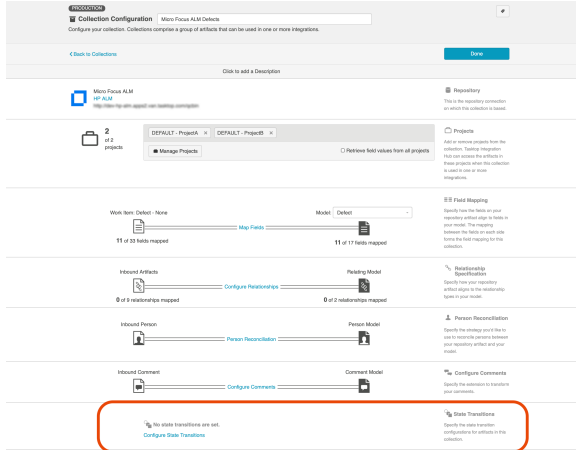
Some repositories require that a state transition be performed in order to update the value of certain fields (for example, when an artifact must move from a status of *New* to *In Progress* to *Closed*, but cannot move directly from *New* to *Closed*). If state transitions are supported for your repository, you will see a State Transition sash at the bottom of the Collection Configuration screen.

You can also review our [Connector Docs](#) to see if state transitions are supported for the repository you are connecting to.

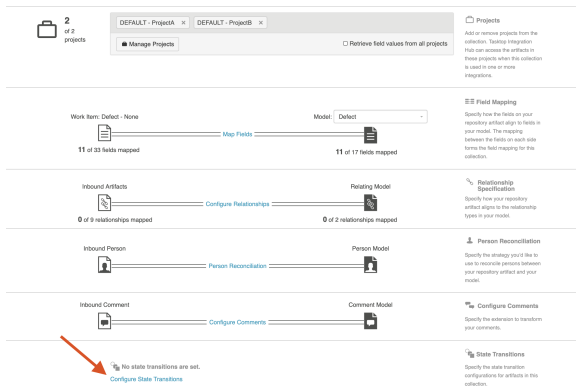
## Configuring State Transitions

If state transitions are supported for your repository, you will see a **State Transition** sash at the bottom of the **Collection Configuration** screen.

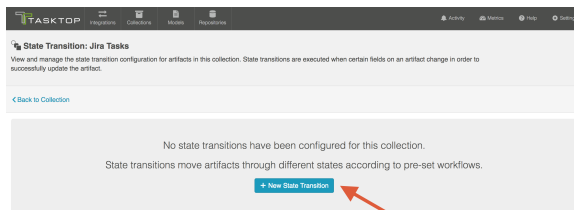
**Note:** State transitions are not supported in Gateway collections.



To set a state transition, click **Configure State Transitions**.

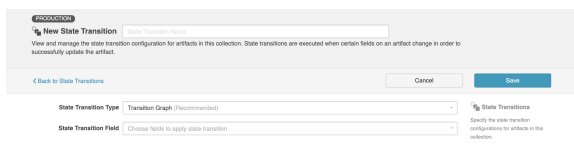


This will lead you to the State Transition screen. Click **+ New State Transition**.



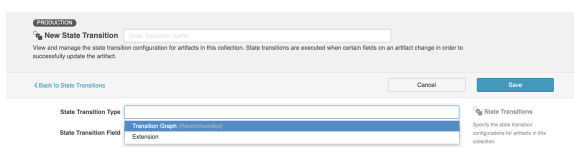
This will lead you to the **New State Transition** screen. Here you can name your transition and choose between two State Transition Types:

- Transition Graph (Recommended)
- Extension

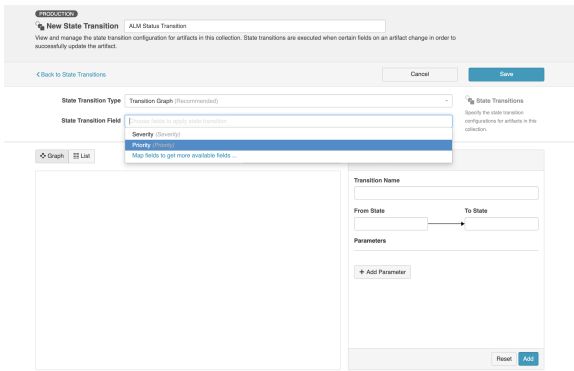


## Transition Graph

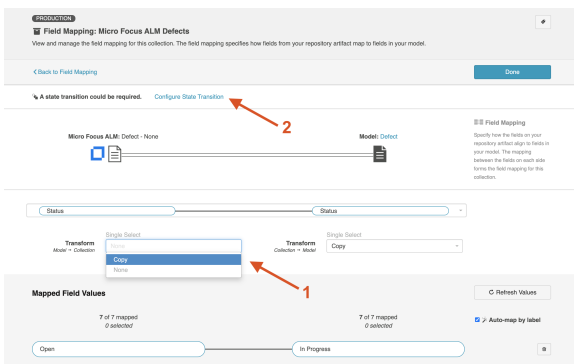
To configure state transitions within Tasktop's UI, select **Transition Graph** as your State Transition Type.



Next, you'll select the **repository** field you'd like to apply the transition to.

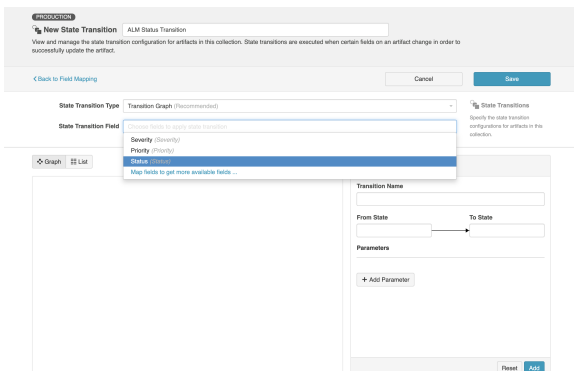


If you don't see the field you'd like to use, make sure that the field is **mapped** and that its **transform is set to copy on the repository side**. Once you set the transform to **Copy**, you will see a **Configure State Transition** link. Click that link to return to the State Transition screen.

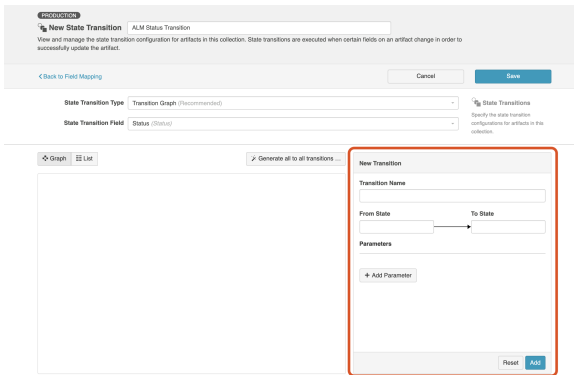


Now you can select the field on the New State Transition screen:

**Note:** If you encounter a **Read-Only** error when selecting the field, please ensure the Field Flow frequency for this field is set to **No Update** on the **Field Flow** screen.



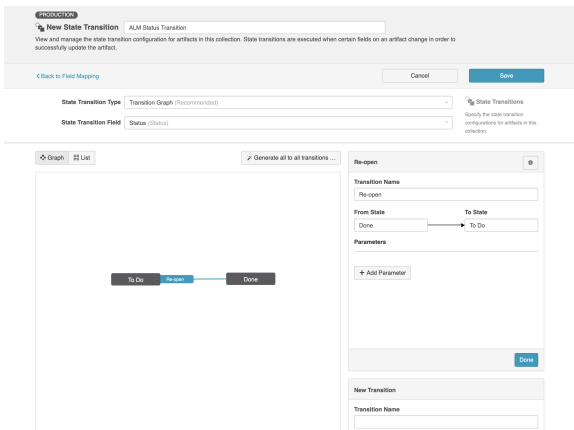
Now that you've selected your field, you'll see the Transition Configuration panel.



You can use the **New Transition** pane to configure your state transitions within Tasktop's UI. In order for your integration to work, these must be configured to match the configuration within the repository itself **exactly**.

When entering values in the **From State** and **To State** fields, the values should match the values within the **repository** (not the model). They must be entered exactly as they appear in the repository, and are both case sensitive and space sensitive. The **Transition Name** must also match the transition name that is configured within the repository exactly.

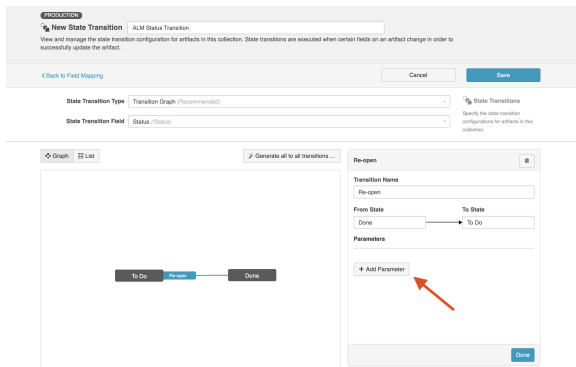
Here is an example of a transition that has been configured. Note that when you view a transition (by clicking on it in the graph), you'll see its configuration on the right so that you can make any needed modifications. You'll also see a **New Transition** pane immediately underneath, so that you can add additional transitions.



**Note:** Multiple transitions between two states in a single direction are not supported. Only a single transition to and/or from two individual states can be configured. Transitions that loop over a single state are also not supported.

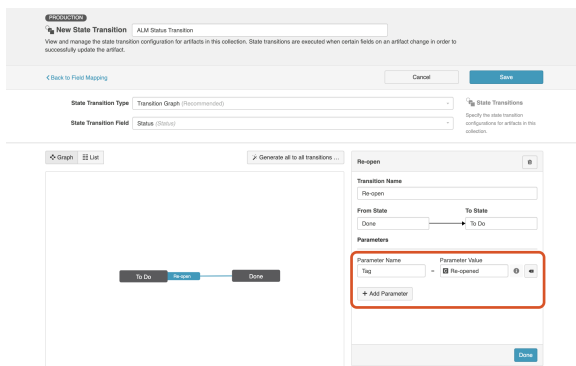
## Adding Parameters

If your transition requires a parameter, click **Add Parameter**.

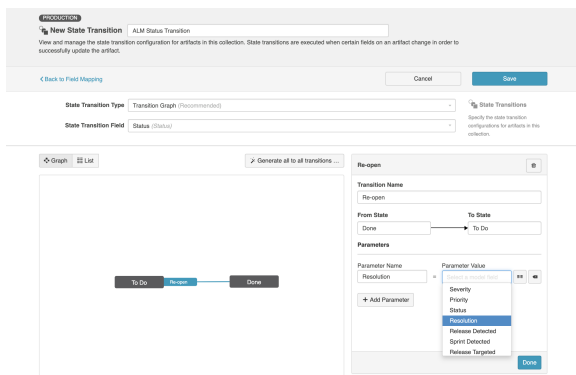


The Parameter name must match the field name within the **repository** exactly. You can either set a constant value for your parameter, or configure the transition to flow a value based on your field mappings.

In the image below, we've set a constant value, which will tell Tasktop to add a **Re-opened** tag to the artifact when it moves through the **Re-open** transition:



You can also set a Parameter that is set based on a field in the **model**.



To map the field, click the **map** icon.

**New Transition**

**Transition Name**  
Move to Done

**From State** → **To State**  
In Progress → Done

**Parameters**

Parameter Name	Parameter Value
Resolution	Resolution

+ Add Parameter

Reset Add

This will bring you to the Parameter Field Value Mappings pop-up.

Parameter Name: Resolution = Parameter Value: Resolution

**Field Value Mappings**

Parameter Field Value:  Add

Model Field Values: Fixed, Deferred, Duplicate

Connect Cancel Apply

Here you can manually enter the parameter field values on the left that exist within your repository, and map them to the model fields on the right. The field values entered must match the field values that exist in the repository **exactly** (they are case- and space- sensitive).

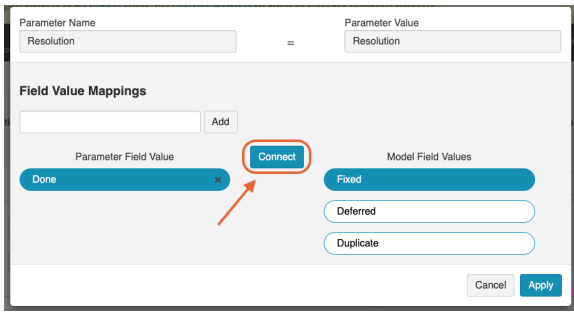
Parameter Name: Resolution = Parameter Value: Resolution

**Field Value Mappings**

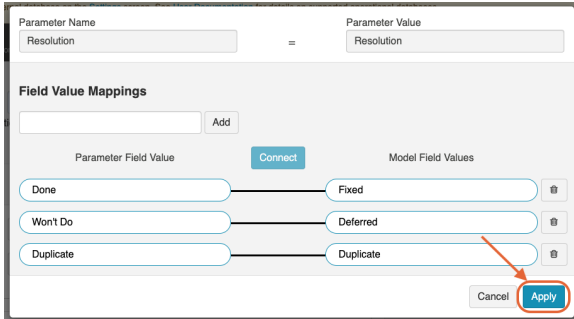
Parameter Field Value: Done Add

Model Field Values: Fixed, Deferred, Duplicate

Connect Cancel Apply



After you have completed mapping your field values, click **Apply** to apply your changes.

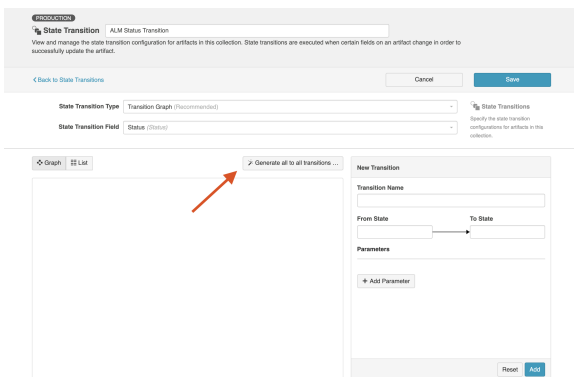


## Generating All to All Transitions

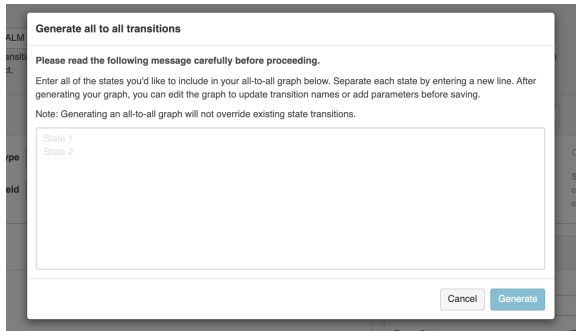
Sometimes workflows require all states to transition to all states and depending on how many states the workflow has, this can result in much manual effort (e.g., for 15 states, you need to create 210 transitions). Rather than manually creating these transitions, Tasktop does most of the work for you by **automatically** generating an all to all state transition graph.

To use this feature, click **Generate all to all transitions**.

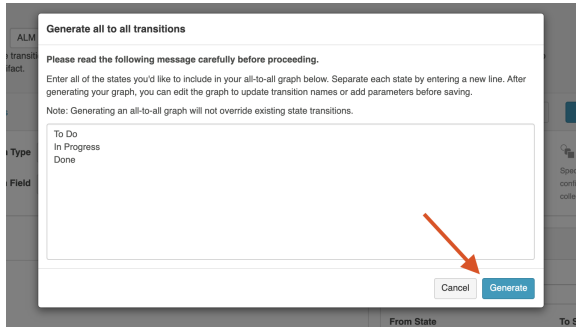
**Note:** Generating an all to all transition graph will not override any existing transitions.



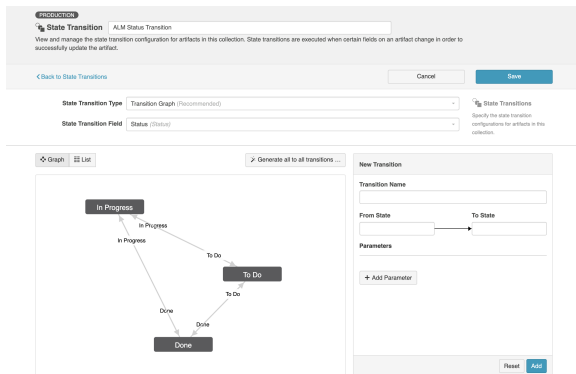
A pop up will appear where you can enter all of the states you'd like to include in your graph.



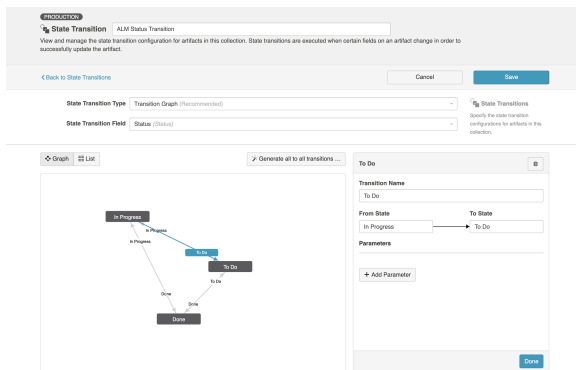
After you've entered all of the states, click **Generate**.



You will then see the automatically generated transition graph with all of the state transitions.



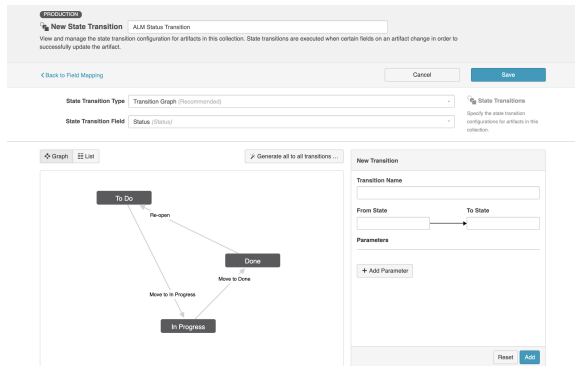
If you'd like to edit a transition, click the transition name and you can edit in the transition configuration panel.





# Saving and Viewing

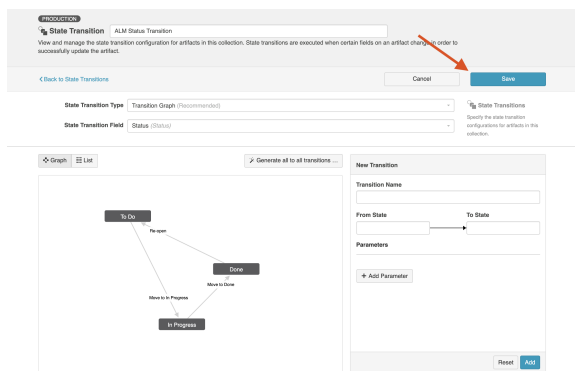
Here's an example of a completed Transition Graph:



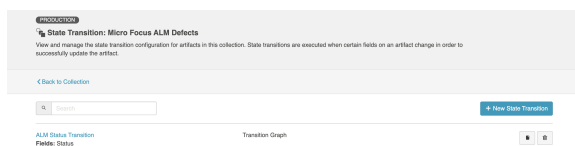
Make sure that your completed graph matches the state transition configuration in your repository **exactly**. If it does not match, you'll see errors when running the integration.

Once confirmed, click **Save** and **Done**.

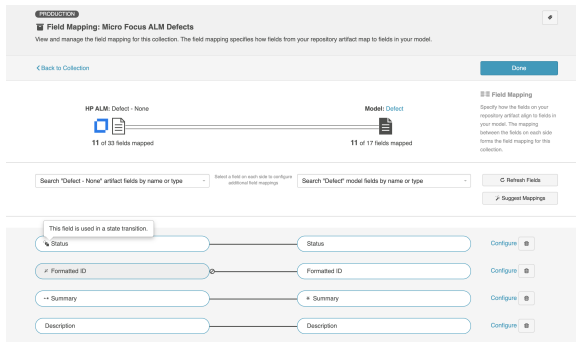
**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.



You can then view or copy your State Transition on the State Transition screen.



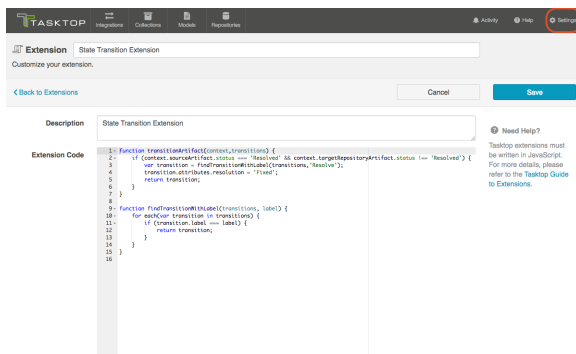
You'll also notice a state transition icon on the collection pill on the Field Mapping screen, denoting that a transition graph is in use.



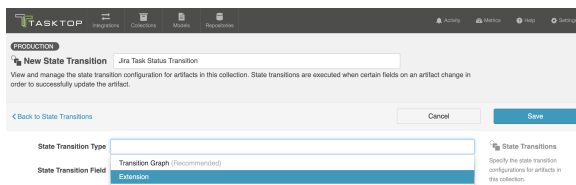
## Extensions

In order to successfully flow field values for fields that require state transitions, a state transition extension can also be set.

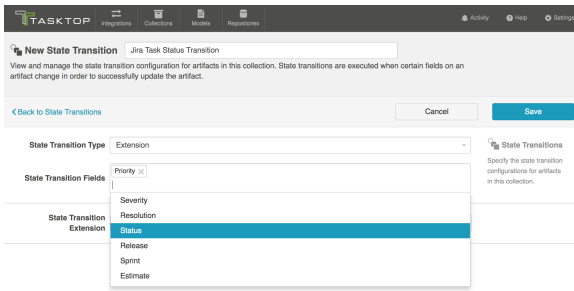
If you choose to configure state transitions via an extension, rather than utilizing the transition graph, your first step will be to create and save the extension itself from the [Extensions \(Settings\)](#) screen. If you need help creating the extension, you can find more information in the [Extensions \(Settings\)](#) section.



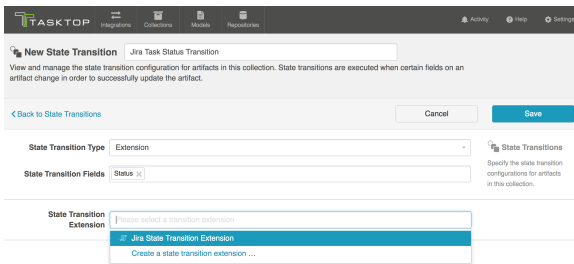
Once the extension is configured, you can select **Extension** as the State Transition Type on the New State Transition screen.



Next, select the **model** field(s) that you'd like to apply the extension to.

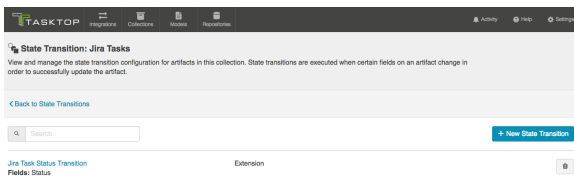


You can then select the extension you'd like to use.

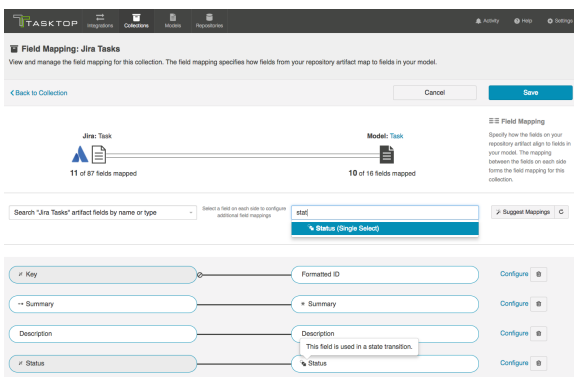


Click **Save** and then **Done**.

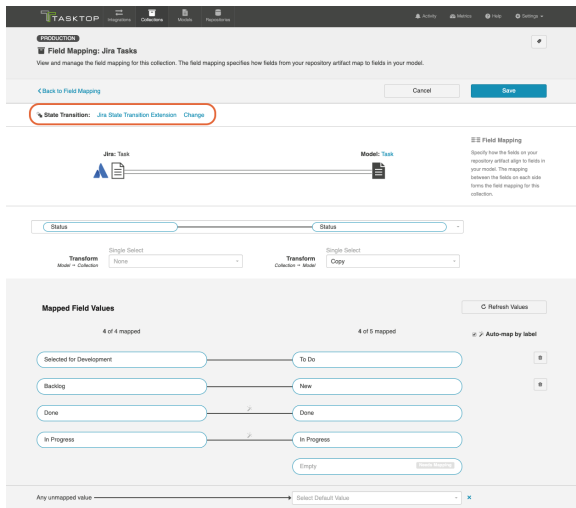
You'll now see the State Transition Extension listed on the State Transition screen.



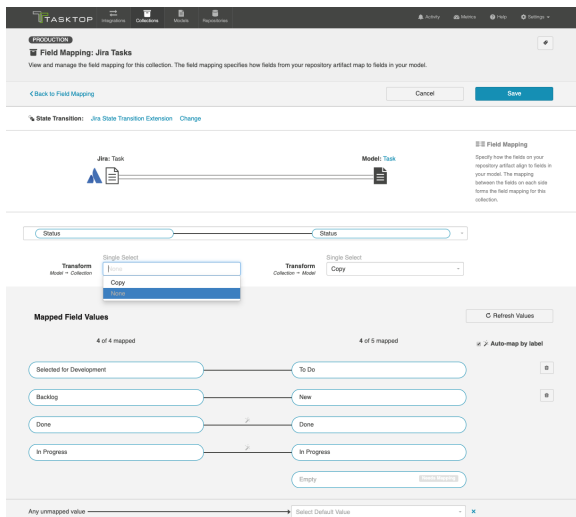
And you'll notice the state transition icon on the model pill and the model drop-down on the Field Flow screen.



You'll also see it listed at the top of the screen when you view the Field Mapping Configuration screen for that field.



**Note:** When using a State Transition Extension, the Transform settings of the Status configuration needs to be set from **Copy** to **None** on the repository side. This can be done in the Field Mapping screen. Click **Save** and then **Done**.



**Note:** The extension will only impact how data flows *from* the model to the repository (Jira in this case). If you would like impact how data flows from the repository to the model (and then to whichever target collection is connected on the other side), you will need to [configure the field appropriately](#). If you would like to use a state transition extension on the other side, you must configure that on the corresponding collection's State Transition screen.

## Next Steps

Once you have completed your State Transition configuration, your next step will be to configure and review your [Artifact Unions](#) if your collection has a single relationship/container field.

# Artifact Unions

## Introduction

Once you have completed your State Transitions configuration, your next step will be to configure and review your Artifact Unions. On this screen, you can specify the related artifact whose fields may flow along with the artifacts in your collection.

In the scenario shown below, the user has a 'Jira Stories to Jama Stories' integration and they'd like the Jira field 'Epic Name' to flow along with the artifacts in their 'Jira Stories' collection, however, the Jira field 'Epic Name' only exists on the 'Epic' artifact.

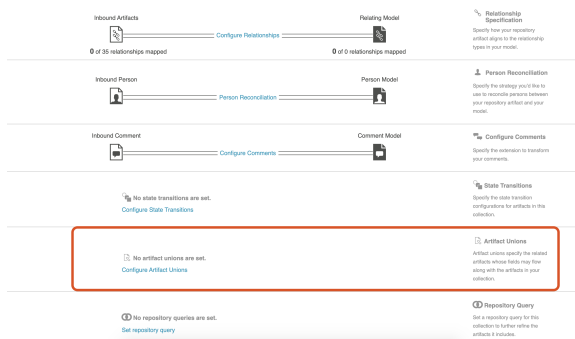
By creating an artifact union, they are able to extract this field from the 'Epic' artifact so that it will flow alongside their Jira Stories collection. After they create the artifact union, it can be mapped to the model where it will flow into their Jama stories collection.

Learn more on how to create and use artifact unions in the sections [below](#).

## Creating Artifact Unions

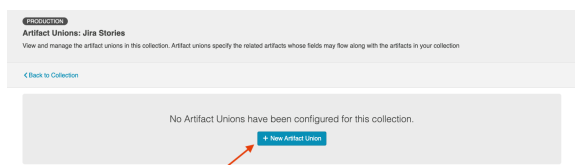
To access your artifact unions, navigate to the **Collection Configuration** screen and click **Configure Artifact Unions**.

💡 The Artifact Unions sash will only be visible when there is a single relationship/container field available for the collection.



To add a new artifact union, click **+ New Artifact Union**

⚠️ While there are no limits on artifact union configurations in a repository collection, please be aware that performance may be impacted if many configurations exist.

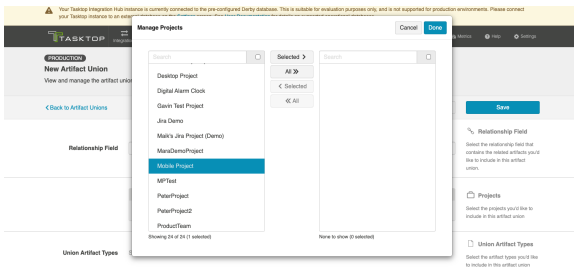


Select the relationship field that links the collection to the target artifact type you'd like to include in your artifact union.


 This dropdown will only contain relationship fields associated with the configured artifact.

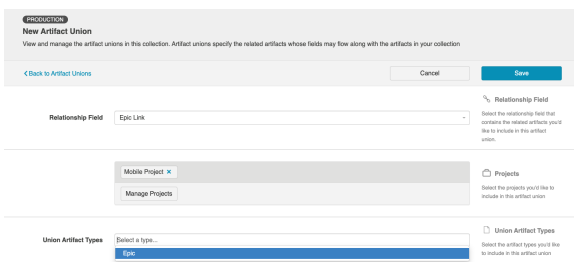


Select the project(s) you'd like to include.



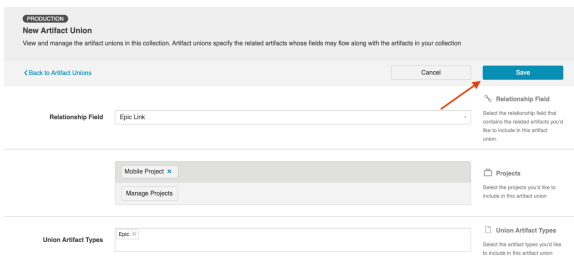
Once you have selected the relationship field and project(s), you can choose the artifact type(s) you'd like to include in your artifact union.

 Only **supported** artifact types will be displayed.



Click **Save** and **Done** to save your artifact union.

 **Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your artifact union.

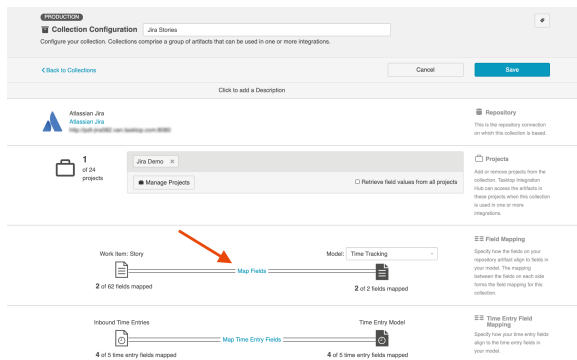


# Utilize your Artifact Unions

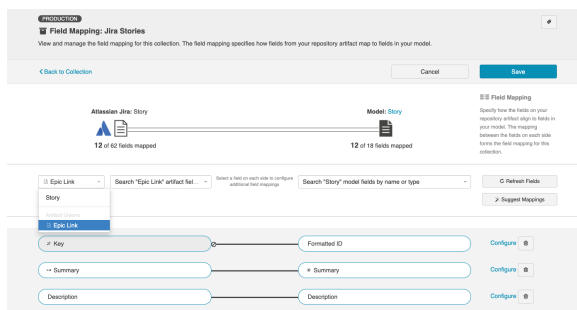
Now that you've created your artifact union, you can now use your artifact union to flow the desired field to the other side of the integration.

💡 Note that fields flowing from artifact unions are read-only.

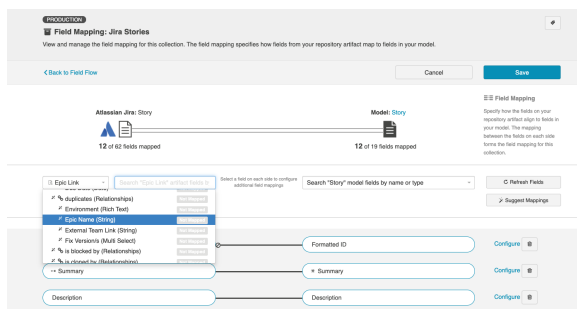
To do this, navigate to the Collection Configuration screen and click **Map Fields**.



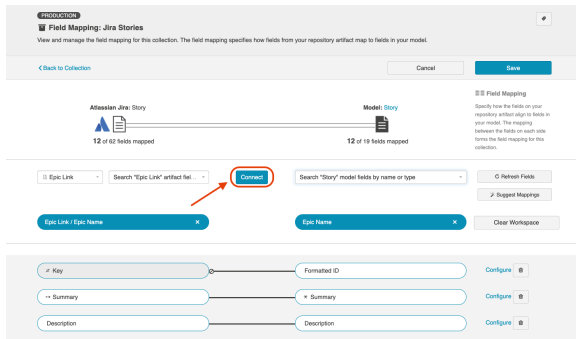
Next, select the configured artifact union from the dropdown menu.



After you've chosen your artifact union, select the field you'd like to extract from the related artifact.

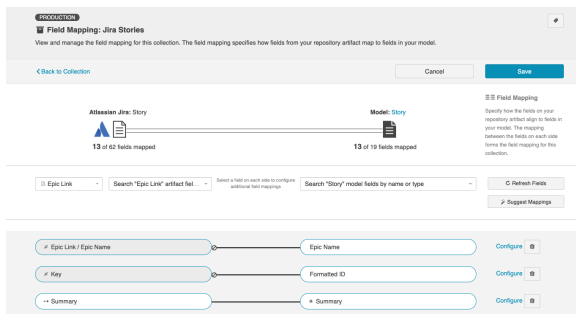


Next, select the model field you'd like to map your field to, and then click **Connect**.



After you've finished mapping your fields, click **Save** and **Done**.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.



That's it! Now, the field will seamlessly flow to the other side of the integration.

## ALM/Octane Test Management

Tasktop Integration Hub offers integration solutions to flow test artifacts such as test results, test steps, and their associated tests, test runs, test instances, and folder structures in Micro Focus ALM and Micro Focus ALM Octane. However, due to complex containment structures and multiple hierarchy levels, sometimes fields and data don't fully align.

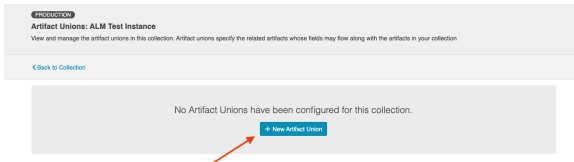
Using artifact unions, you can easily flow test artifacts and their related fields and sub-entities from ALM to Octane.

In the example below, the user has an 'ALM Test Instance to Octane Tests' integration and they'd like to flow 'Test Steps' from ALM into their Octane Tests collection. However, ALM Test Steps only exist on the ALM 'Test' artifact. Because the ALM Test Instance has a shared relationship field with ALM Tests, they are able to configure an artifact union between ALM Test Instances and ALM Tests so that the Test Steps will flow into their Octane Tests collection.

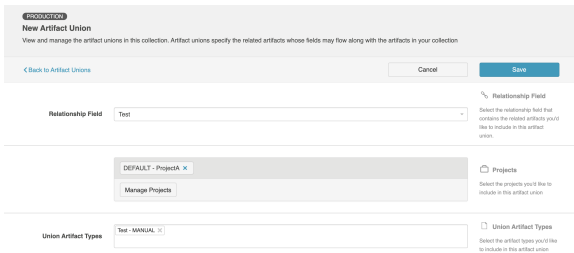
### Example

First, the user navigates to their ALM Test Instance collection and creates a new artifact union.

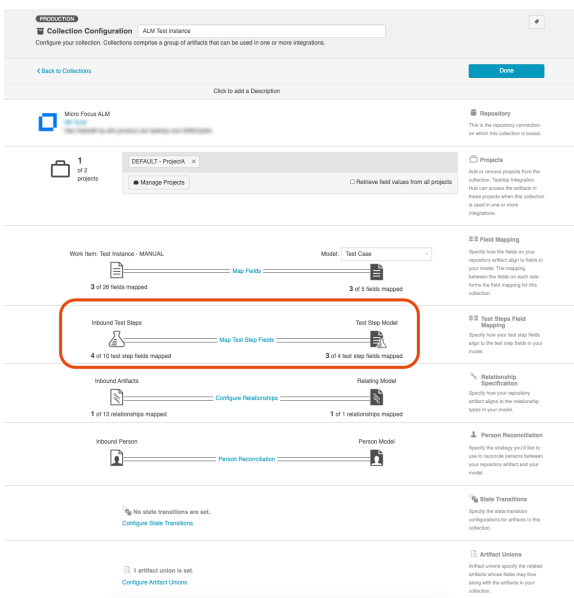




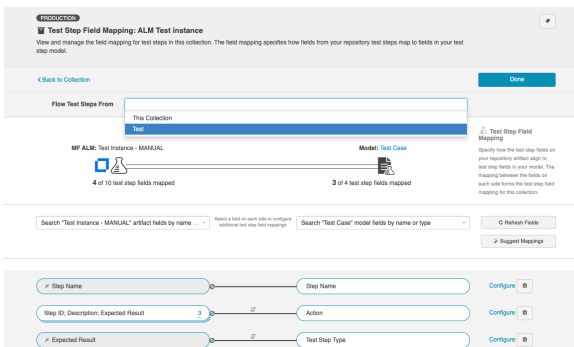
Then, the user selects the **Test** relationship field, their desired project, and artifact type.



After saving the artifact union, the user clicks **Map Test Step Fields** on the Collection Configuration screen.



On this screen, the user selects the **Test** artifact union from the **Flow Test Steps From** dropdown menu.



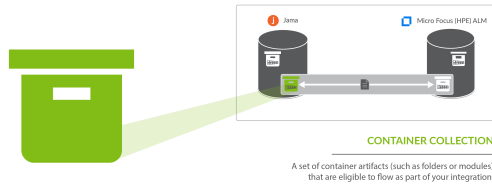
After saving, the user can flow the ALM test steps into their Octane tests collection. And that's it!

## Next Steps

Now that your Artifact Unions are configured, your collection configuration is complete. Once all the collections you'd like to utilize in your integration are set up, it's time to move on to [Step 4: Configure your Integration](#) .

# Container Collection (Repository)

## What is a Collection?



You can think of a *collection* as the set of artifacts that are eligible to flow as part of your integration. The process of creating a collection consists of a few steps which whittle down your repository into a smaller subset of artifacts. To create your collection, you will specify:

1. The repository the artifacts live in
  - a. Each collection can only come from *one* repository
2. The artifact type (i.e. defect, folder, etc)
  - a. Each collection can only contain *one* artifact type
3. The projects within the repository those artifacts live in
  - a. Each collection can contain one or multiple projects
4. The model you would like your collection to be mapped to (not pictured)
  - a. Each collection can be mapped to one and only one model



You can learn more about collections in the [Key Concepts](#).

## What is a Container Collection?

There are two types of repository collections:

- **Work Item Collections**, which include 'work items' used to track development work. These are artifacts such as defects, requirements, or test cases.
- **Container Collections**, which include 'containers' used to organize your work. These are artifacts such as folders, modules, and packages. Containers are used to organize work items into groups.

On this page, we will be showing you how to configure a **Container Collection**.

## Video Tutorial

Check out the video below to learn how to configure a Container Collection.

## Configuring a Container Collection

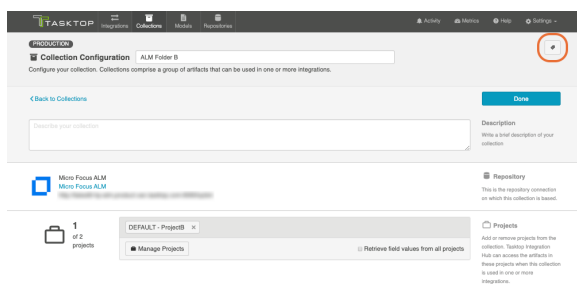
The steps to configure a Container Collection are very similar to the steps to configure a [Work Item Collection \(Repository\)](#). Please refer to that page for in depth instructions.

You will, however, notice a few key differences:

- After clicking *New Collection*, you will select *Container Collection*, instead of *Work Item Collection*.
- The artifact type selected for a container collection, must be a **container**, such as a folder, module, or package. Some repositories may be ineligible for container collections, as they may not include the appropriate artifact types. Consult our [Connector Docs](#) to see which container types are supported for each repository.
- When you create a container collection, you'll notice that the model selected defaults to the out-of-the-box Container model. This will allow you to take advantage of built-in Smart Fields, which will auto-map to your collection.
- Container collections will typically have fewer fields to map than a work item collection.
- It is generally very important to map the 'parent' field for a container collection. This will enable you to preserve the correct hierarchical relationships between your containers when flowing them to a target repository. If you are using the out-of-the-box Container model, Tasktop will be able to auto-map this for you in most scenarios.
- Container collections typically will not contain a 'status' field, and therefore will not require state transition mappings.

## Viewing Associated Configuration Elements

To view associated configuration elements (such as models or integrations that utilize the collection you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



 Associated Elements for Repository Collection "ALM Folder B"

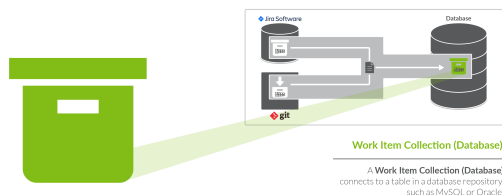
- 1 Model used by this Repository Collection
  - [Simple Container](#)
- 1 Repository Connection used by this Repository Collection
  - [Micro Focus ALM](#)
- ≡ 1 Integration using this Repository Collection
  - [ALM Containment Mirroring Synchronizations](#)

Close

# Work Item Collection (Database)

Database Collections are only available in Editions that contain the Enterprise Data Stream add-on. See [Tasktop Editions table](#) to determine if your edition contains this functionality.

## What is a Work Item Collection (Database)?



There are two types of Work Item Collections: Repository Collections, which connect to repositories like *Jira* or *Micro Focus ALM* and Database Collections, which connect to databases, such as *MySQL*. On this page, we will be teaching you how to configure a database work item collection.

A Database Work Item Collection connects to a table in a database repository, such as *MySQL* or *Oracle*. Once your Database Work Item Collection is configured, you can flow information from artifacts in your source collections (either Repository or Gateway Collections) to that table, via an Enterprise Data Stream Integration.

You can learn more about collections in the [Key Concepts](#).

## Video Tutorial

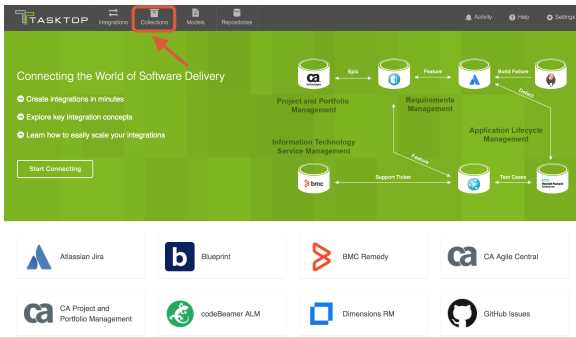
Check out the video below to learn how to create a new collection for your database repository:

**Note:** In version 18.1 and later, you will select 'Work Item Collection' as your template, rather than 'Repository Collection' as shown in the video.

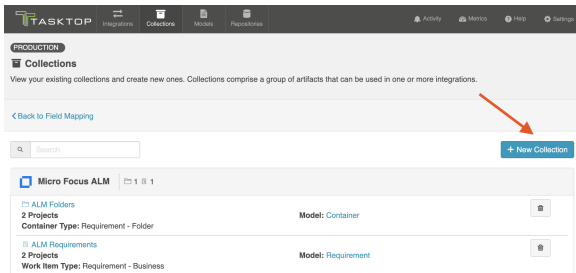
## Creating a Database Collection

To create a database work item collection, follow the steps below.

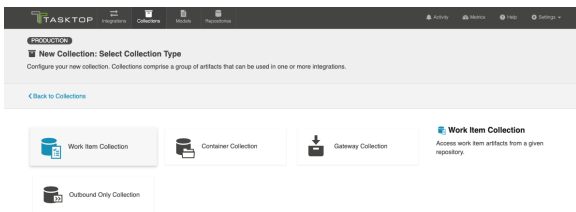
Select **Collections** at the top of the screen.



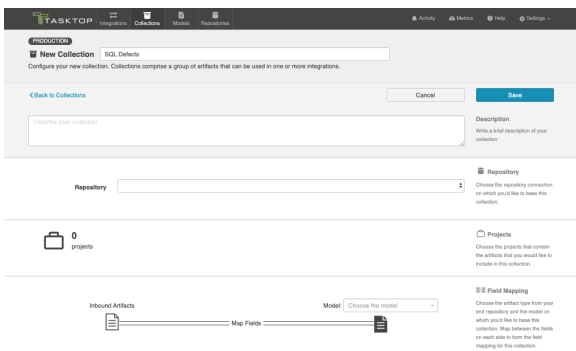
Click **New Collection**.



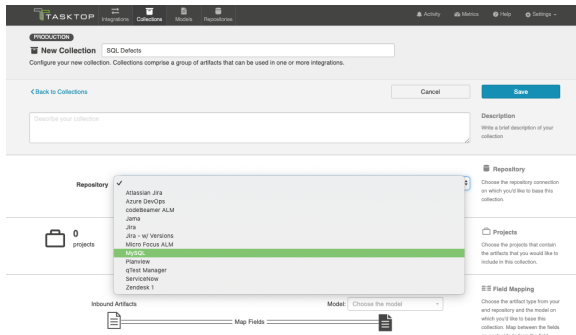
Select **Work Item Collection** as the collection type.



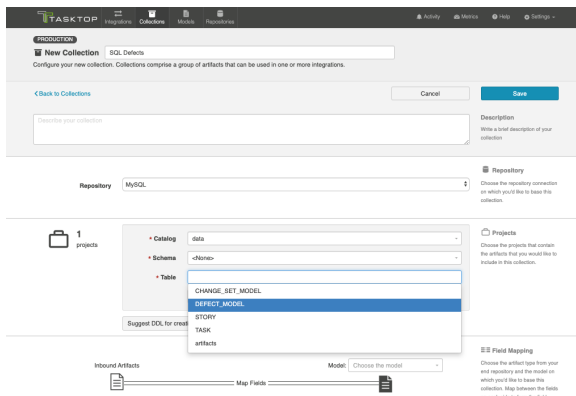
Enter the name for your collection.



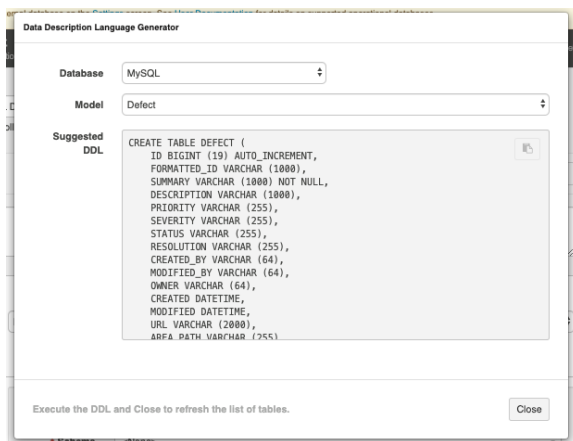
Select the Connection on which you'd like to base this collection. In our example, we are selecting MySQL, which is the **Tasktop SQL** repository connection we have configured.



Select the database table that will receive artifacts that flow to this collection.

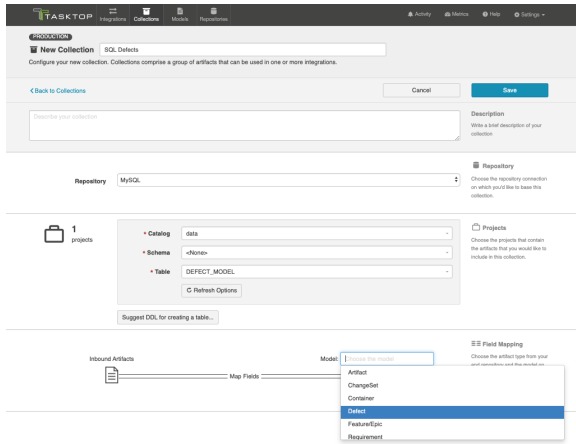
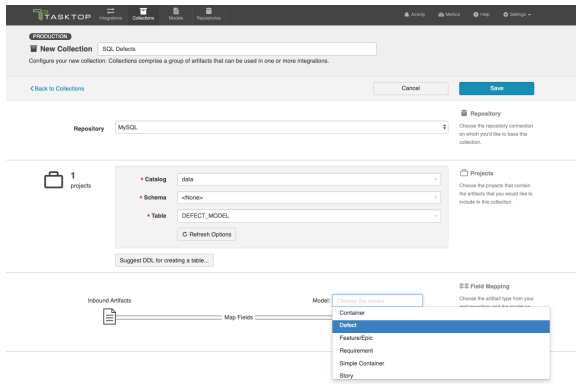


**Note:** if your table is not listed, you can use the **Suggest DDL** tool to generate a SQL command that can help you create a table that aligns with the model on which you'd like to base this collection.



Select the model on which you'd like to base this collection.

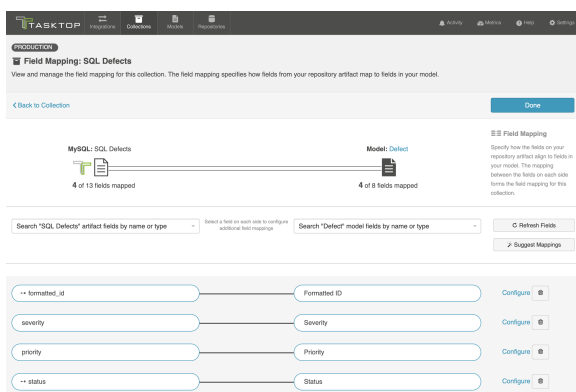




## Map Fields

Now that you have identified the model, you can complete the collection-to-model field mapping by going into the **Map Fields** link.

**Note:** If you used the Suggest DDL tool to create your database table, the mapping will be done automatically.



## Constant Value Mapping

In some scenarios, the database might require that some of its columns/fields always have a value. This value is usually provided by mapping it to the equivalent model field. When there is no equivalent field

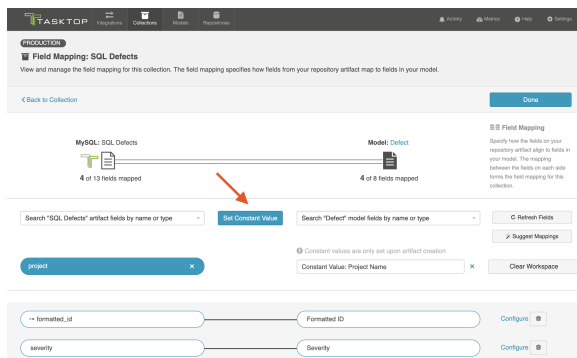
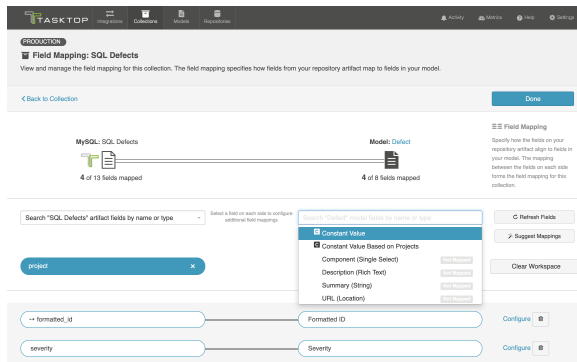
in the model that can provide a value, you can set a constant value into your end-database column /field. The value you configure will then always get written out.

To set a constant value for a field, select the **Constant Value** option from the drop down menu on the model side. This will tell the integration to *always* flow that value to the database collection. Enter the value, and then click the **Set Constant Value** box.

**Note:** Constant values can be set for the following fields types:


- Boolean
- Date/DateTime
- Double
- Location
- Long
- Multi Select
- Person
- Rich Text
- Single Select
- String

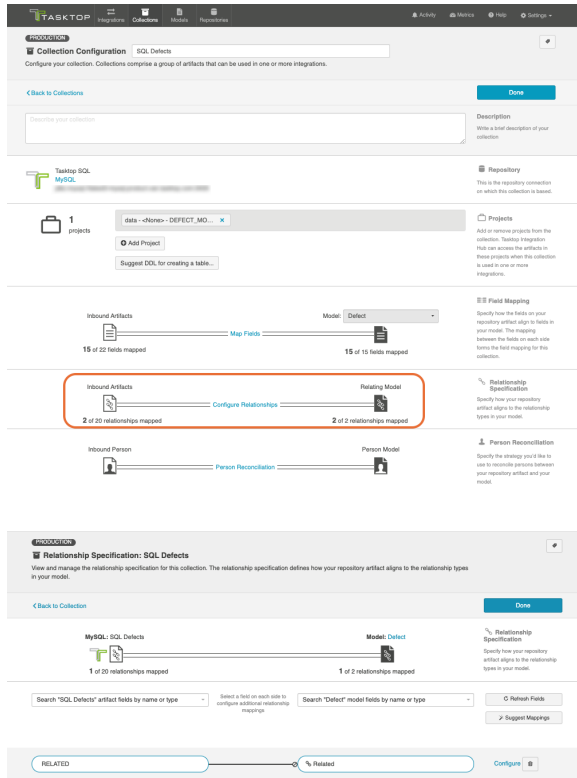
Only some of these types are relevant for your database collection, however, given the field types that can be configured in the database itself.



## Configure Relationships

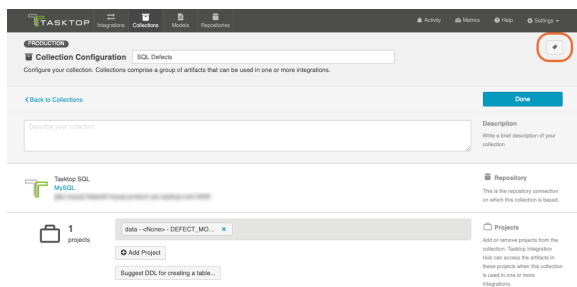
If you have any relationship(s) fields in your model, you can map those on the **Configure Relationship Types** screen of a given collection.

 **Note:** if you used the Suggested DDL tool to create your database table, the mapping should be done generally.



## Viewing Associated Configuration Elements

To view associated configuration elements (such as models or integrations that utilize the collection you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



 Associated Elements for Repository Collection "SQL Defects"

■ 1 Model used by this Repository Collection

- [Defect](#)

■ 1 Repository Connection used by this Repository Collection

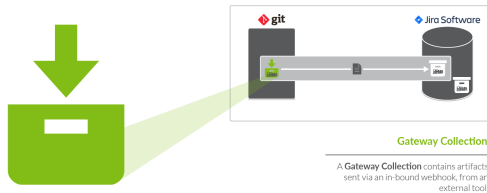
- [MySQL](#)

Close

# Gateway Collection

## What is a Gateway Collection?

*Gateway Collections are only available in Editions that contain the Gateway add-on. See [Tasktop Editions table](#) to determine if your edition contains this functionality.*



You can think of a *collection* as the set of artifacts that are eligible to flow as part of your integration. A **gateway collection** contains artifacts sent via an in-bound webhook, from a DevOps tool.

You can learn more about collections in the [Key Concepts](#).

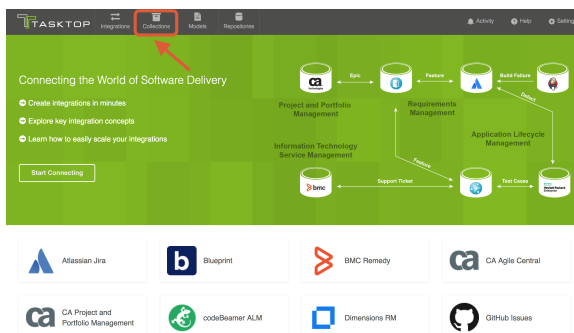
## Video Tutorial

Check out the video below to learn how to create a new gateway collection:

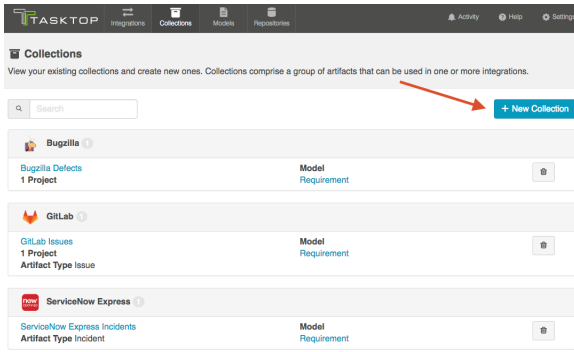
## Creating a Gateway Collection

To create a gateway collection, follow the steps below.

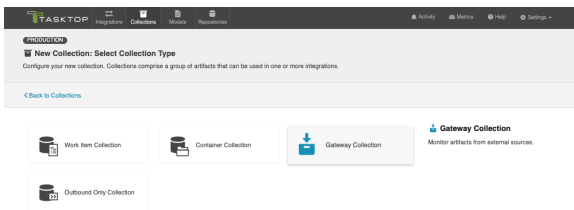
Select **Collections** at the top of the screen:



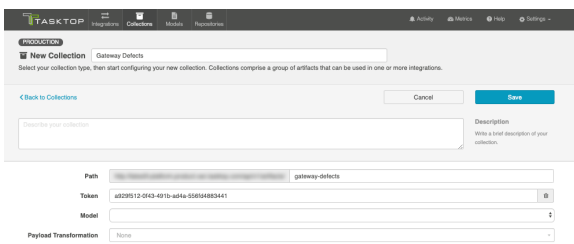
Click **New Collection**.



Select **Gateway Collection** as the collection type.

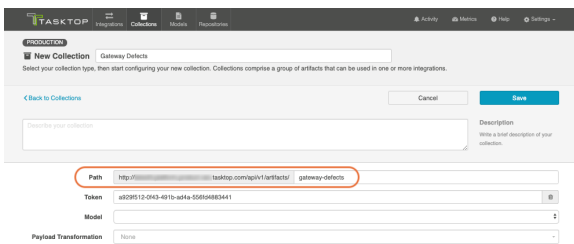


Enter a name for your collection.



Next, specify the **path** for your collection. These characters will form the REST endpoint to which you can send artifacts to Tasktop via this gateway collection.

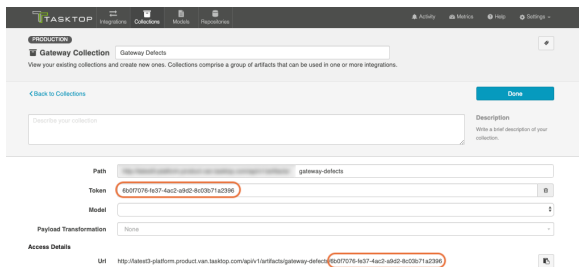
💡 Upon first creating your gateway collection, Tasktop will populate path with the name that you have given to your collection. You can change this if desired.



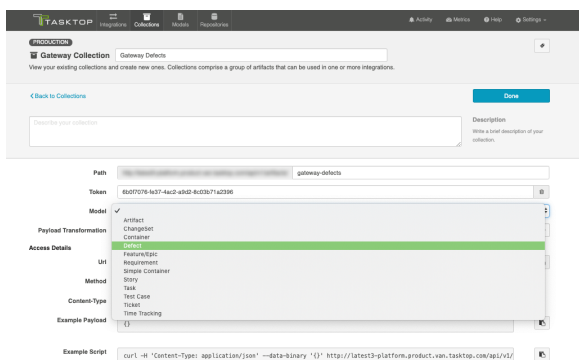
To **secure your gateway collection**, Tasktop automatically appends a token (a universally unique identifier) to the path of a gateway collection. This token will be incorporated into your gateway URL and will help ensure that only users that know the full path with its token can access your gateway collection.

If using Tasktop On-prem, you can remove the token by clicking the trash can icon to the right, and refresh it by hitting the magic wand icon that appears in its place (for security, tokens cannot be removed on Tasktop Cloud). Once refreshed, click **Save**, and the URL will be updated.

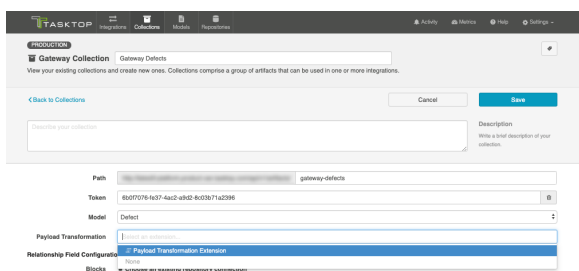
**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your collection.



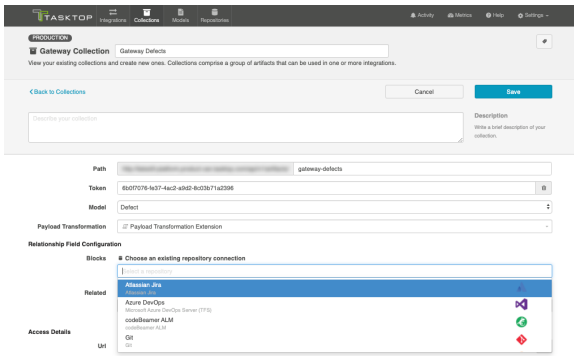
Select the model on which you'd like to base the collection.



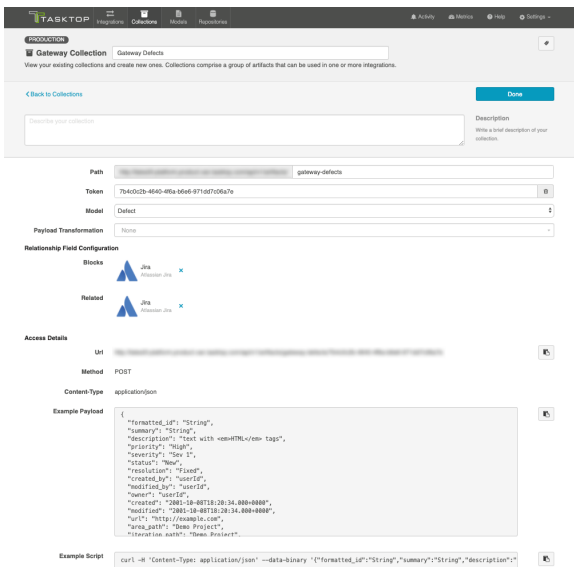
If you have configured a [payload transformation extension](#) for your gateway collection on the Extensions (Settings) screen, you can select it here.



Once you click **Save** you'll notice that additional fields appear. If you have any relationship(s) fields in your model, you'll need to identify a target repository for each. This will ensure that enough information is being sent in via the gateway to uniquely locate the artifact you'd like to relate to.



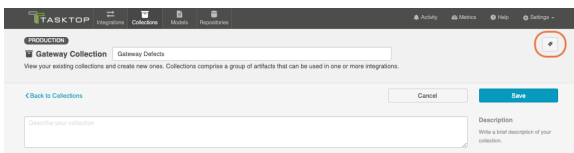
Once you've saved your collection, you will be able to observe the access details given for this gateway collection.



The example payload can be used to construct the JSON payload that will be sent to Tasktop from your external tool.

## Viewing Associated Configuration Elements

To view associated configuration elements (such as models or integrations that utilize the collection you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.





 Associated Elements for Gateway Collection "Gateway Defects"

■ 1 Model used by this Gateway Collection

- [Defect](#)

≡ 2 Integrations using this Gateway Collection

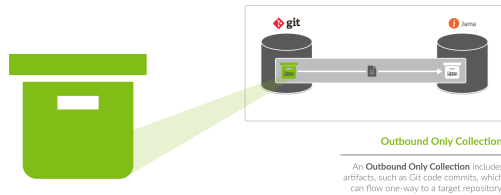
- [Defect Reporting](#)
- [Jira Defect Creation](#)

Close

# Outbound Only Collection

## What is an Outbound Only Collection?

*Outbound Only Collections are only available in editions that have access to the Git repository.*



You can think of a *collection* as the set of artifacts that are eligible to flow as part of your integration. An **outbound only collection** contains artifacts like code commits or changesets, which you may want to flow out of your repository, but which would not receive updates into your repository.

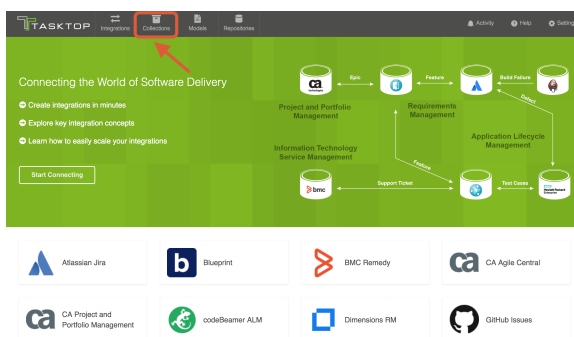
You can learn more about collections in the [Key Concepts](#).

**Note:** Outbound Only collections can connect to the Git repository only. You can learn more about configuring that repository in our [Connector Docs](#).

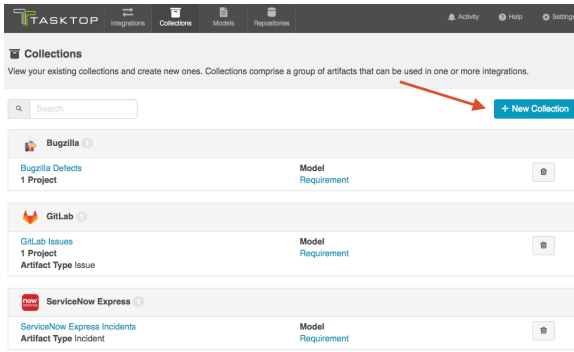
## Creating an Outbound Only Collection

To create an outbound only collection, follow the steps below.

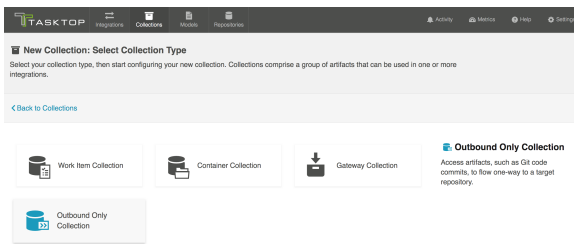
Select **Collections** at the top of the screen.



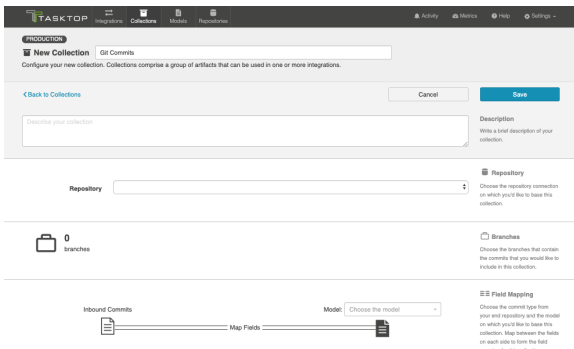
Click **New Collection**.



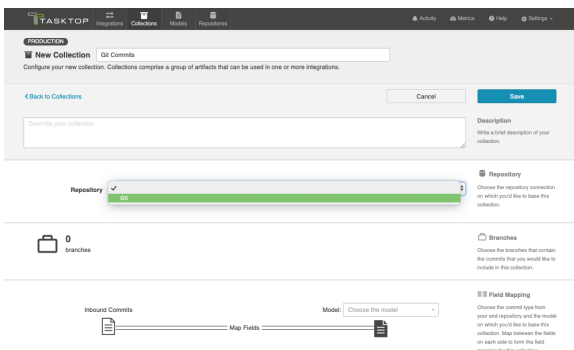
Select **Outbound Only Collection** as the collection type.



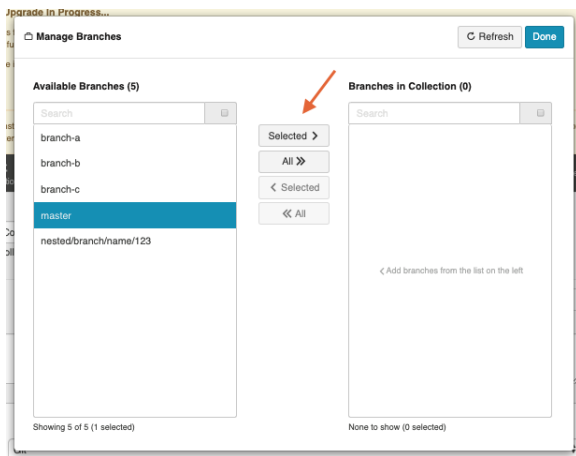
Enter a name for your collection.



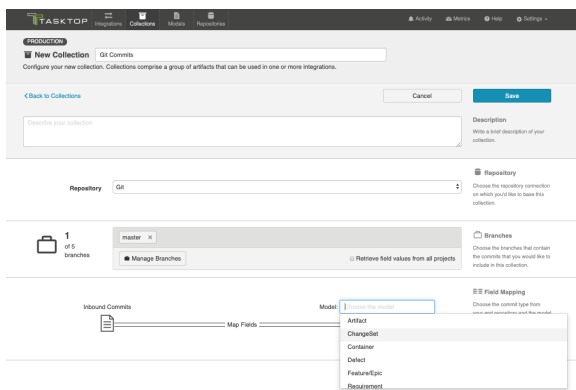
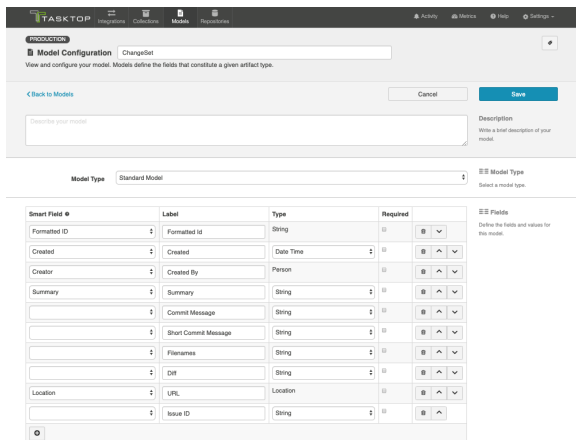
Select the repository that you would like to connect. The Outbound Only collection type can only connect to the Git repository. The collection will include artifacts from the repository you have selected.



Add branches to your collection by selecting **Manage Branches**. These are the branches from which Tasktop will be flow code commits, changesets, or other artifacts.

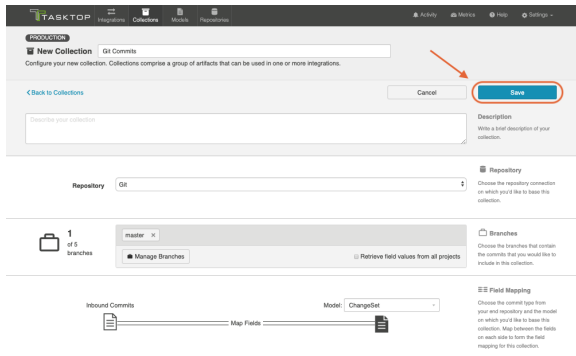


Select the model you'd like to use for this collection. If available, choose the ChangeSet model. We recommend ensuring that your model contains the following fields:

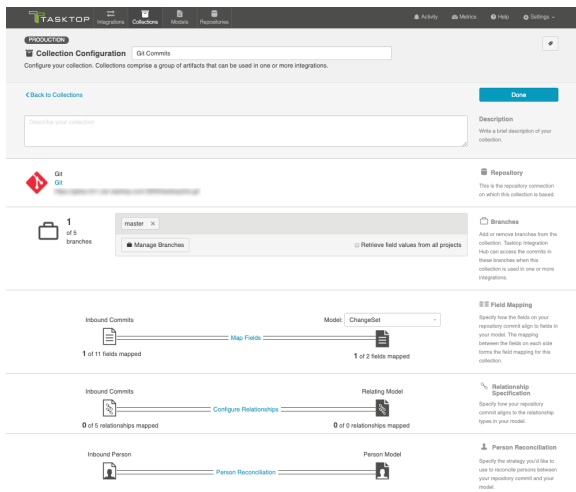


Click **Save**.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your collection.



After you have saved your collection, you will be able to configure your collection.

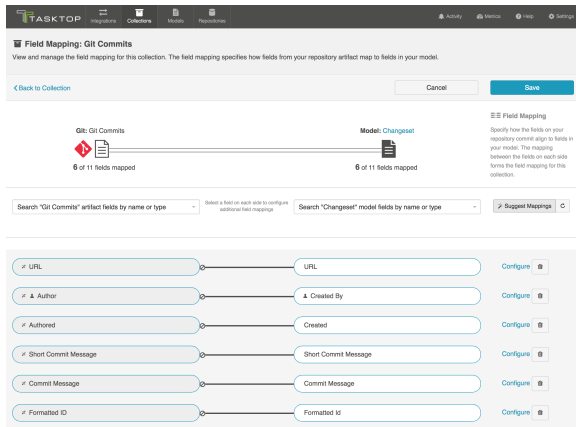


## Map Fields

Clicking **Map Fields** will take you to the Field Mapping screen. On this screen, you will be able to specify how fields in your repository are mapped to fields in your model. You'll notice that for an Outbound Only collection, fields can only flow in one direction: out of your collection, and into your model. This mapping will determine how information flows from fields in your source collection to fields in your target collection.

You can learn more about the Field Mapping screen [here](#).

Here's an example field mapping configuration for a Git Commit collection:



## Configure Relationships

Clicking **Configure Relationships** will take you to the Relationship Specification screen. On this screen, you will be able to specify how **relationship** fields in your repository are mapped to fields in your model. Relationship fields, such as 'blocked by,' 'is related to,' and 'parent,' enable you to preserve the relationship structure between artifacts as you flow information from one collection to the other.

You can learn more about this process on the [Relationship Specification](#) page.

## Person Reconciliation

Clicking **Person Reconciliation** will take you to the Person Reconciliation screen. On this screen, you will be able to specify the strategy you'd like to use to reconcile person fields between your repositories.

You can learn more about this process on the [Person Reconciliation](#) page.

## Optional: Set a Repository Query

If you have enabled repository queries for the repository that you have connected to, you will also see a **Repository Query** sash at the bottom of the screen.

**⚠** Note that Repository Queries are advanced functionality, and should only be used when you are truly unable to filter as desired using the built-in Tasktop functionality of Repositories, Collections, and Artifact Filtering.

You can learn more about Repository Queries [here](#).

## Viewing Associated Configuration Elements

To view associated configuration elements (such as models or integrations that utilize the collection you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.

**TASKTOP** | Home | Collections | Actions | Repositories | Activity | Metrics | Help | Settings

**COLLECTIONS**

**Collection Configuration** | Git Commits

Configure your collection. Collections comprise a group of artifacts that can be used in one or more integrations.

[Back to Collections](#) **Done**

**Description**  
Write a brief description of your collection.

**Git**  
 **Git**  
 This is the repository connection on which this collection is based.

**Repository**  
 This is the repository connection on which this collection is based.

**Branches**  
 **Branches**  
 Add or remove branches from the collection. Tasktop Integration will use the settings in these branches when this collection is used in one or more integrations.

**1** **Git**

**master**  **Retrieve field values from all projects**

**Manage Branches**

**Associated Elements for Repository Collection "Git Commits"**

- 1 Model used by this Repository Collection**
  - [ChangeSet](#)
- 1 Repository Connection used by this Repository Collection**
  - [Git](#)

Close






# Step 4: Configure your Integration

## Types of Integration Templates

Tasktop offers a range of integration templates to enable you to achieve a diverse set of goals:

## Integration Styles

Build a custom integration based on one of these integration styles.




 <p><b>Work Item Synchronization</b></p>	 <p><b>Container + Work Item Synchronization</b></p>	 <p><b>Create via Gateway</b></p>	 <p><b>Modify via Gateway</b></p>	 <p><b>Enterprise Data Stream</b></p>
<p><i>The Work Item Synchronization template is available in all Editions.</i></p>	<p><i>The Container + Work Item Synchronization template is available in all Editions.</i></p>	<p><i>The Create via Gateway template is only available in Editions that contain the Gateway add-on. See <a href="#">Tasktop Editions table</a> to determine if your edition contains this functionality.</i></p>	<p><i>The Modify via Gateway template is only available in Editions that contain the Gateway add-on. See <a href="#">Tasktop Editions table</a> to determine if your edition contains this functionality.</i></p>	<p><i>The Enterprise Data Stream template is only available in Editions that contain the Enterprise Data Stream add-on. See <a href="#">Tasktop Editions table</a> to determine if your edition contains this functionality.</i></p>
<p>This integration connects teams</p>	<p>This integration connects teams working in</p>	<p>This integration creates traceability between artifacts across the software development</p>	<p>This integration creates traceability between artifacts across the software development lifecycle. Already existing</p>	<p>This integration simplifies enterprise reporting by unlocking software lifecycle data from its application tool silos and</p>



<p>working in different tools as they fulfill their roles in the software development lifecycle.</p> <p>As part of this integration, work items will flow between separate repository collections.</p>	<p>different tools as they fulfill their roles in the software development lifecycle.</p> <p>As part of this integration, work items will flow between separate repository collections.</p> <p>Additionally, the containers in which your work items reside will be mirrored across the collections according to your specification.</p>	<p>lifecycle. New artifacts will be created in a repository collection when artifacts are sent to Tasktop via a Gateway collection, through an inbound webhook.</p>	<p>artifacts in a repository collection will be located and modified in a specified way when artifacts are sent to Tasktop via a Gateway collection, through an inbound webhook.</p>	<p>providing a rich data repository for near real-time analytics. Records will be created in a single database when artifacts from one or more collections are created or changed.</p>
<p><a href="#">Learn More</a></p>	<p><a href="#">Learn More</a></p>	<p><a href="#">Learn More</a></p>	<p><a href="#">Learn More</a></p>	<p><a href="#">Learn More</a></p>

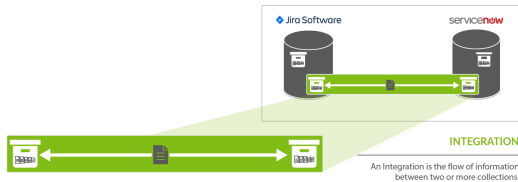
## Prebuilt Integration Patterns

Base your integration on one of these patterns to address typical business needs.

 <p style="text-align: center;"><b>Code Traceability: Create and Relate a Changeset</b></p>	 <p style="text-align: center;"><b>Code Traceability: Update an Existing Work Item</b></p>	 <p style="text-align: center;"><b>Test Synchronization</b></p>
<p><i>This integration template is only available in editions that have access to the Git repository.</i></p>	<p><i>This integration template is only available in editions that have access to the Git repository.</i></p>	<p><i>See the <a href="#">Tasktop Editions table</a> to determine if your edition contains this functionality.</i></p>
<p>This integration creates new work items such as changesets or code commits in a repository such as Jama, when they are sent to Tasktop via a read-only collection connecting to a repository such as Git.</p> <p>These types of events are “fire and forget” - they create something new in your repository, but they don’t expect anything back. As such, they don’t mandate a full-blown two-way synchronization; a lighter one-way integration can do the trick.</p>	<p>This integration flows information from an outbound only collection (such as Git Commits) to a field on an existing artifact in a work item collection (such as Jama Codes).</p> <p>These types of events are “fire and forget” - they create something new in your repository, but they don’t expect anything back. As such, they don’t mandate a full-blown two-way synchronization; a lighter one-way integration can do the trick.</p>	<p>This set of integrations flows Test Result information on a Test Run in ALM or Tosca.</p> <p>To ensure proper test artifact hierarchy is preserved within each tool, Test Folders, Tests, Test Set Folders, Test Sets, and Test Instances are also synchronized.</p>
<p style="text-align: center;"><a href="#">Learn More</a></p>	<p style="text-align: center;"><a href="#">Learn More</a></p>	<p style="text-align: center;"><a href="#">Learn More</a></p>

# Work Item Synchronization

## What is an Integration?



An *integration* is quite simply **the flow of information between two or more collections**.

A *work item synchronization* is a specific type of integration that flows **work items** (such as defects, requirements, or stories) between two **repositories**.

When you configure your work item synchronization, you can customize the field flow, artifact routing, artifact filtering, as well as enable or disable comment flow or attachment flow.

## Video Tutorial

Check out the video below to learn how to configure a Work Item Synchronization.

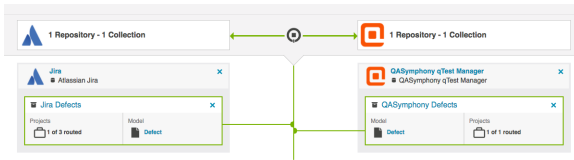
⚠ This video assumes that you have already configured your repositories, models, and collections as outlined in the [Quick Start Guide](#).

## Use Case and Business Value

The Work Item Synchronization Template connects teams working in different tools as they fulfill their roles in the software development lifecycle. It allows you to flow work items (such as defects or requirements) from one repository to the other.

As part of this integration,

- Work Items, such as defects or requirements, will flow between two work item (repository) collections.
- Artifact Creation Flow can be configured either one-way or two-way
- You'll also configure the direction and frequency in which each field on those artifacts, as well as comments and attachments, should be updated.



## Template Affordances

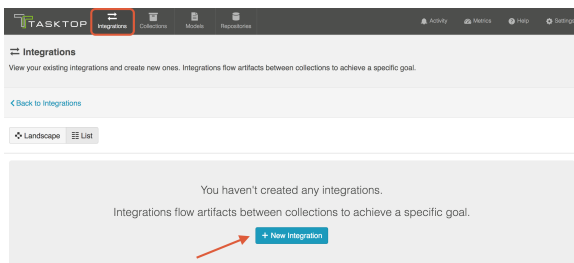
The Work Item Synchronization template allows you to flow artifacts between two work item (repository) collections.



## Configuring a Work Item Synchronization Integration

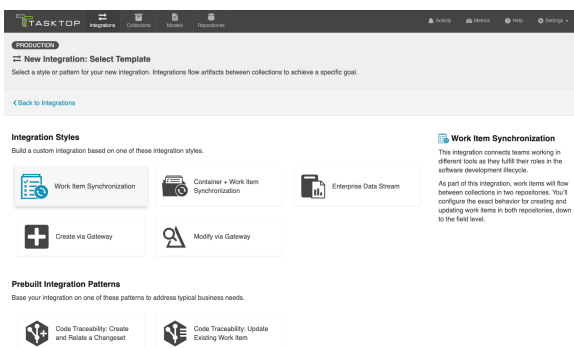
Now that you have all of your base components (i.e., repositories, models, and collections) set up, you can configure an integration to synchronize the artifacts in your collections.

To configure your integration, select **Integrations** at the top of the screen, then click **+ New Integration**.

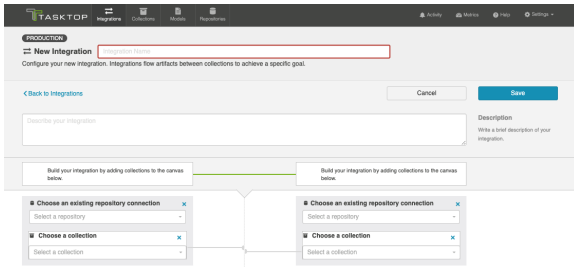


Select the **Work Item Synchronization** template.

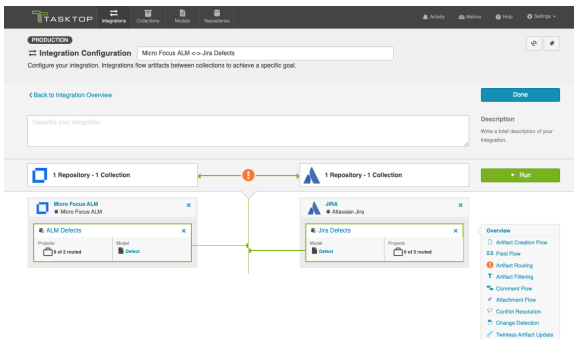
💡 Depending on the **edition** of Tasktop you are utilizing, you may not have all options available.



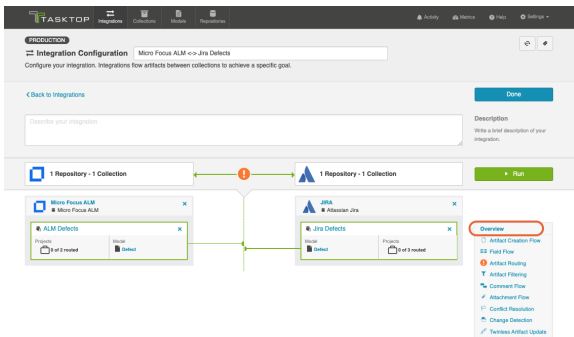
This will bring you to the **New Integration** screen.

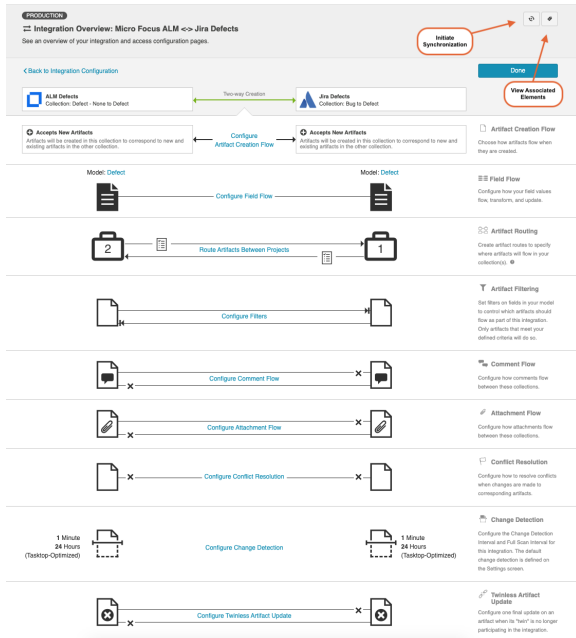


Name and describe your integration, and select your repositories and collections.



You can click the **Overview** link on the right side of the Integration Configuration screen to get to the main display page (shown in the second screen shot).





From this page, you can configure many different components of your work item synchronization.

## Artifact Creation Flow

On the **Artifact Creation Flow** screen, you can specify whether new artifacts will be created in one collection or both. You can learn more on the [Artifact Creation Flow](#) page.

## Field Flow

On the **Field Flow** screen, you can configure how your field values will flow, transform, and update between each collection. Each field can be configured individually. You can learn more on the [Field Flow](#) page.

## Artifact Routing

On the **Artifact Routing** screen, you can specify where (in which projects) new artifacts will be created, based on the projects they originate from in the source collection. You can learn more on the [Artifact Routing](#) page.

## Artifact Filtering

On the **Artifact Filtering** screen, you can set filters on fields in your model to control which artifacts will flow as part of the integration. Only artifacts that meet your defined filter criteria will be eligible to flow. You can learn more on the [Artifact Filtering](#) page.

## Comment Flow

On the **Comment Flow** screen, you can enable or disable comment flow. You can learn more on the [Comment Flow](#) page.

## Attachment Flow

On the **Attachment Flow** screen, you can enable or disable attachment flow. You can also set a maximum attachment size limit. You can learn more on the [Attachment Flow](#) page.

## Test Step Flow

Depending on your [Tasktop edition](#), you may see a **Test Step Flow** sash. You can learn more about this feature on the [Test Synchronization](#) page.

## Conflict Resolution

On the **Conflict Resolution** screen, you can set a strategy to determine how to resolve conflicts when changes are made to both the source and target artifact. You can learn more on the [Conflict Resolution](#) page.

## Change Detection

On the **Change Detection** screen, you can set custom change detection and full scan intervals for your integration. Change Detection and Full Scan intervals define how frequently Tasktop will search for updates made to artifacts in each repository. The settings configured here will override the default global change detection settings configured on the [General \(Settings\)](#) screen. You can learn more on the [Change Detection](#) page.

## Twinless Artifact Update

On the **Twinless Artifact Update** screen, you can configure one final update (for example a comment or a status change) on an artifact when its "twin" in the other repository is no longer eligible to participate in the integration (for example when it's been deleted or no longer meets the artifact filter). The final update informs the newly twinless artifact that the synchronization has been discontinued.

This feature demystifies the integration process and allows end users to understand why an artifact may no longer be receiving updates via the Tasktop integration. Once notified of the change via a comment or field update on the artifact, users can work with their Tasktop admin or with users in the other system to troubleshoot. You can learn more on the [Twinless Artifact Update](#) page.

## Initiate Synchronization

You will also notice an **Initiate Synchronization** button in the upper right corner of the screen. This button can be used to immediately initiate synchronization for selected projects participating in your integration. This is beneficial if artifact filters are expanded, making it such that new artifacts are eligible for integration. You can learn more [here](#).

## Viewing Associated Configuration Elements

To view associated configuration elements (such as collections or models that utilize the integration you are viewing), click the **Associated Elements** tag in the upper right corner of the screen next to the **Initiate Synchronization** button.



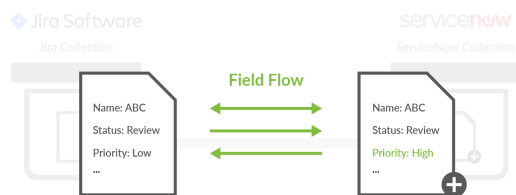
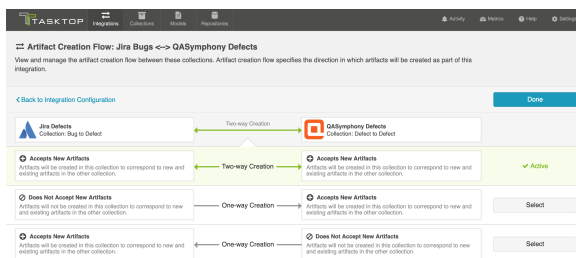
# Artifact Creation Flow

## Introduction

After saving your [Work Item Synchronization](#), the next step is to configure Artifact Creation Flow. Artifact Creation Flow specifies whether new artifacts will be created in one repository or in both.

Note that Artifact Creation Flow refers only to the **creation** of artifacts (as opposed to the updating of fields on those artifacts). So for example, if you set up one-way *artifact creation flow* from Jira to ServiceNow, this means that when the integration is run, new or existing artifacts from Jira will create new artifacts in ServiceNow, but new or existing artifacts from ServiceNow will **not** create new artifacts in Jira.

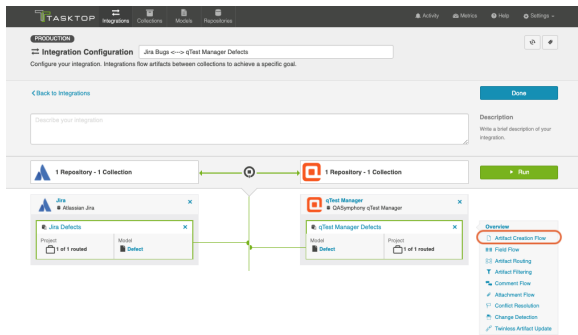
However, once a Jira artifact creates a target artifact in ServiceNow, if any updates are made to fields on the target artifact in ServiceNow, those updated fields **could** flow back over to Jira, based on the integration's [field flow configuration](#). So while the integration is not **creating** new artifacts in Jira, it can **modify** existing artifacts in Jira based on updates made to the corresponding artifacts in ServiceNow, depending on how the field flow has been configured in Tasktop.



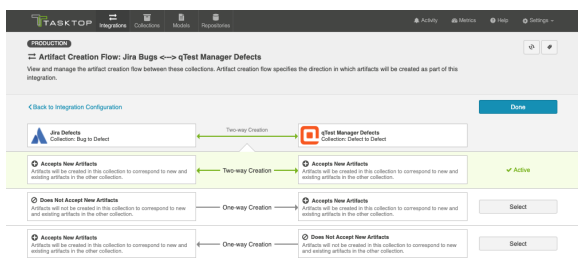
Note that **Field Flow** is set independently for each field pair, and does not need to match the configuration for **Artifact Creation Flow**. In the example above, if the priority on our ServiceNow artifact is changed from Low to High, that updated field value will flow back to Jira.

## Configuring Artifact Creation Flow

To configure Artifact Creation Flow, click the **Artifact Creation Flow** link on the **Integration Configuration** screen.



This will lead you to the Artifact Creation Flow screen, where you will be able to select Two-way Creation (artifacts will be created in both collections to correspond to new and existing artifacts in the other collection), or One-way Creation (only one of the two repositories will have new artifacts created to correspond to new and existing artifacts in the other collection).



Click **Save** and **Done**.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.

You will then be brought back to the Integration Configuration screen.

## Next Steps

Once you have completed your Artifact Creation Flow configuration, your next step will be to review your [Field Flow](#).

# Field Flow

## Introduction

Once you've configured your [Artifact Creation Flow](#), your next step is to configure your Field Flow.

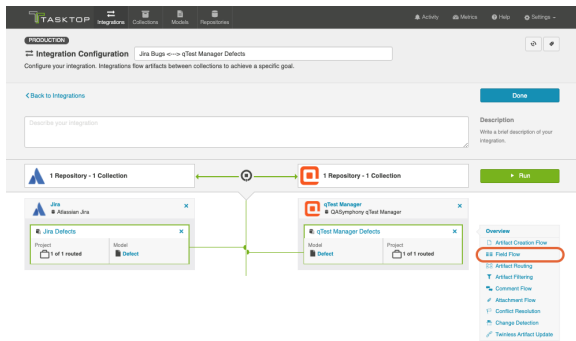
On the Field Flow screen, you can configure:

- the direction fields flow in
- the frequency with which they flow (i.e., only upon creation vs. always updating)
- how your field values will flow under set conditions

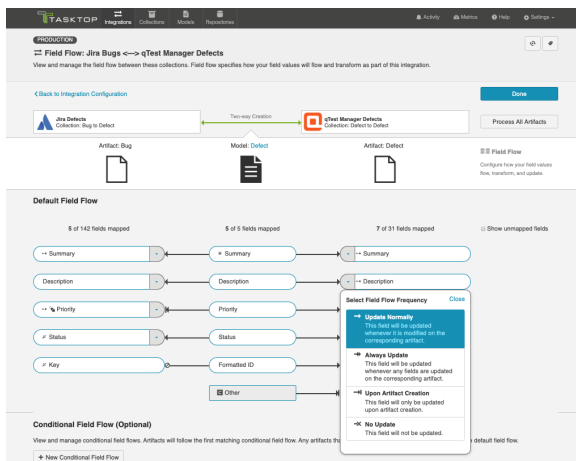
Each field can be configured individually.

## Configuring Field Flow

To get to the Field Flow screen, click the **Field Flow** link on the **Integration Configuration** screen.



You will be directed to the **Field Flow** screen.



Here, you can see the names of the mapped repository fields for each collection on the far left and right, with the model fields displayed in the middle. By default, model fields without mapped repository fields are hidden. You can see all model fields by toggling the **Show unmapped fields** checkbox. Constant values will be identified by a grey box and the constant value icon.

Once you're done updating your field flow, click **Save** and **Done**.

 **Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.

## Conditional Field Flow

See [Tasktop Editions table](#) to determine if your edition supports Conditional Field Flow functionality.

When ownership of an artifact changes during the artifact lifecycle, data conflicts can occur if field values are updated on both sides. With **Conditional Field Flow**, you can avoid these conflicts by setting conditions on the field flow of an artifact.


You can use conditional field flow to determine:

- **Which** fields will flow from one tool to the other
- **How** the fields should flow (i.e., direction of the field flow)

To use a field for conditional field flow, it must:

- Be a part of your model, and be mapped in each collection
- Be one of the following field types:
  - Single Select
  - Multi-Select
  - Boolean
  - Date
  - Date/Time
  - Duration
  - String
  - Long
  - Double

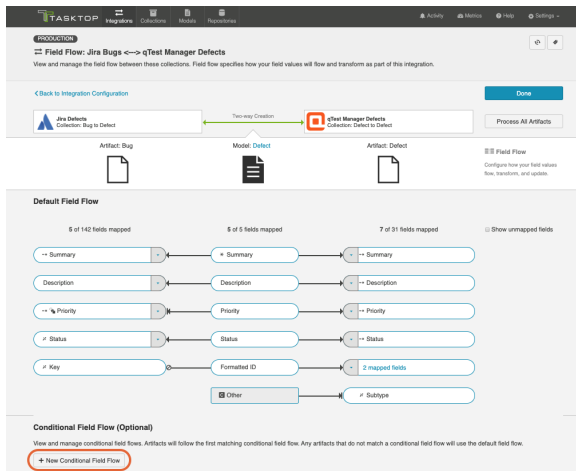
You can create **multiple** conditional field flows, which will be executed in order from top to bottom.

 **Note:** If an artifact does not meet the set condition(s), the default field flow will be used. If the artifact meets multiple conditions, the first matching conditional field flow will be used.

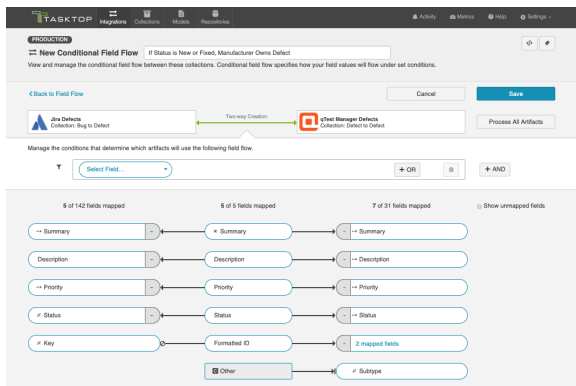
In the scenario configured [below](#), ownership of an artifact changes from the manufacturer (Jira) to the supplier (qTest) when the status of the artifact is updated from **New** to **In Progress**, and back to the manufacturer when updated to **Fixed**. To prevent conflicting data, conditional field flows are used to only flow field values from the owner of the artifact when these changes occur.

## Configuring Conditional Field Flow

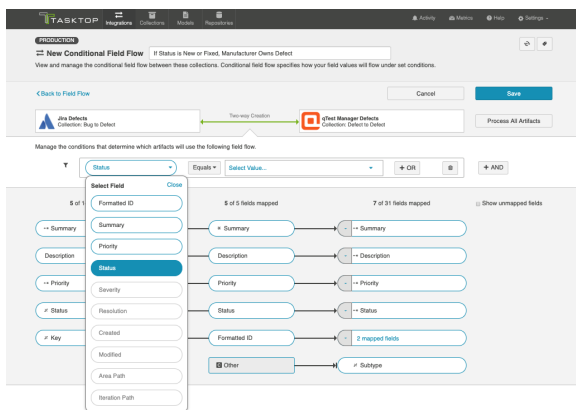
To configure a conditional field flow, click **New Conditional Field Flow**.

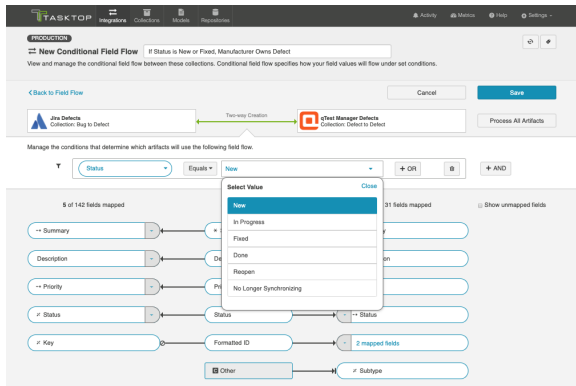


This will bring you to the **Conditional Field Flow** screen.



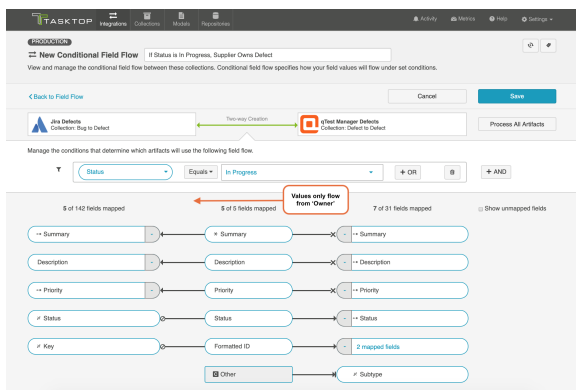
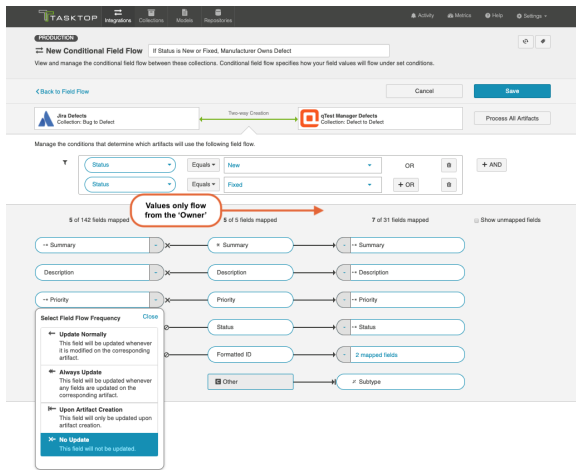
Select a field and set your desired conditions.



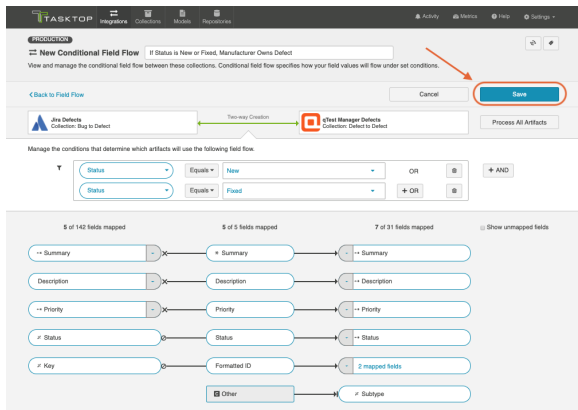


Select your field flow frequency to determine the direction of your field flow.

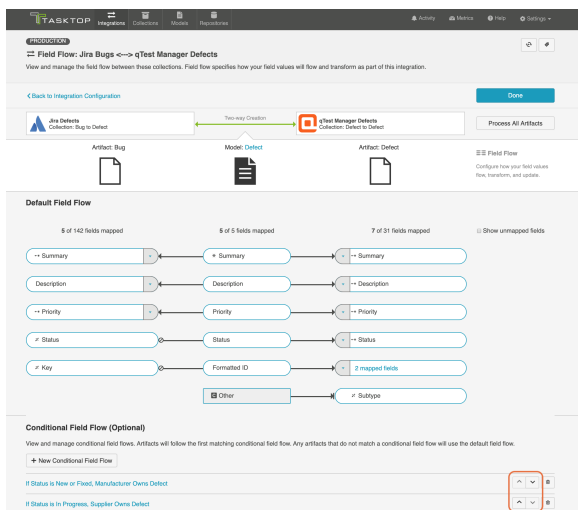
In this scenario, the field flow frequency is set to **No Update** so that the field values will only flow from the *manufacturer* (Jira). If the *supplier* (qTest) owns the defect, field values will only flow from the *supplier* (see second screenshot).



After you have finished configuring your conditional field flow(s), click **Save** and **Done** to save your changes.



Once you have saved your conditional field flow(s), they will appear on the Field Flow screen and can be re-ordered by using the arrows.





**Note:** If multiple changes are made to an artifact in an interval of time shorter than the integration's change detection interval, these changes will be detected all at once and the field flow will be chosen based on the state of the artifact at the time it was detected by Hub.

## Field Flow Direction and Frequency

When configuring field flow for a synchronization integration, you have several options available to specify the direction and frequency of field updates:




Icon	Meaning
→	<b>Update Normally:</b> This field will be updated whenever it is modified on the corresponding artifact
⇒	<b>Always Update:</b> This field will be updated whenever <u>any</u> fields are updated on the corresponding artifact

	<p> <b>Note:</b> "Always Update" can only be used concurrently with "No Update" or "Upon Artifact Creation" field flow frequencies. If attempting to use "Always Update" with "Update Normally" or "Always Update" on both sides, a conflict will occur.</p>
→	<b>Upon Artifact Creation:</b> This field will only be updated upon artifact creation
→X	<b>No Update:</b> This field will not be updated










 **Note:** The field flow settings behave a bit differently for Constant Values. This is because constant values exist as part of your Tasktop configuration, and not on the artifact itself. Therefore, changes in constant values are not detected in the same way that updates made on the actual artifact are detected. If you change the constant value that is linked to your model, your integration will not automatically detect this update and sync it over. The value will only update if another field on that artifact is updated. Because of this, for constant values, "update normally" and "always update" will behave identically: meaning that the constant value will update whenever any other field is updated on that artifact.

## Field Flow Icons

On the Integration Field Flow screen, you will see a number of icons, which will help you understand any special properties or requirements for each field. If you hover your mouse over an icon, you will see a pop-up explaining what that icon means. You can also review their meanings in the legend below:


Icon	Meaning
	<p>A constant value will be sent.</p> <p>Note that:</p> <ul style="list-style-type: none"> <li>• If the icon is on the side of the collection, this means that a constant value will be sent to your model. This means that any time this collection is integrated with another collection, the <i>other</i> collection will receive this constant value for the field in question.</li> <li>• If the icon is on the side of the model, this means a constant value will be sent to your collection. This means that any time this collection is integrated with another collection, that <i>this</i> collection will receive this constant value for the field in question.</li> </ul>
	<p>A state transition will be utilized. Note that:</p> <ul style="list-style-type: none"> <li>• If the icon is on the side of the collection, this means that a <a href="#">state transition graph</a> is being utilized.</li> <li>• If the icon is on the side of the model, this means that a <a href="#">state transition extension</a> is being utilized. You can determine which collection it applies to based on whether it is left-aligned or right-aligned.</li> </ul> <p> Note that Tasktop will update the field flow frequency for fields utilizing state transitions to 'no update.' This is because they are updated via the transition and not via normal 'field flow.' Do not modify the field flow frequency for this field.</p>



	Collection field is read-only and cannot receive data
 	To create artifacts in your collection, this field must be mapped to your model.
	This is a required field in your model; it must be mapped to your collection.
	This field will not be updated as part of your integration, due to how you have configured it. This field flow configuration can be changed if you'd like.
	This field will not be updated as part of your integration because the mapping would be invalid. You do not have the option of changing this.
	This field will update normally as part of your integration; this means it will be updated whenever it is modified on the corresponding artifact.
	This field will always update as part of your integration; this means that it will be updated whenever any fields are modified on the corresponding artifact.
	This field will only be updated upon initial artifact creation.

## Process All Artifacts

The **Process All Artifacts** button will prompt Tasktop to process all artifacts in the integration. Any changes or additions you've made to your collection-to-model mappings will be applied to all artifacts participating in the integration upon the next change detection interval. This functionality can be useful when adding a new field to your field flow configuration. You can learn more about this process [here](#).

 **Note:** Process All Artifacts will attempt to re-synchronize all mapped fields on all artifacts. If there are some fields in your collections that are only writeable upon creation, this may cause errors. To avoid this, ensure that all such fields have their Field Flow set to update **Upon Artifact Creation**.

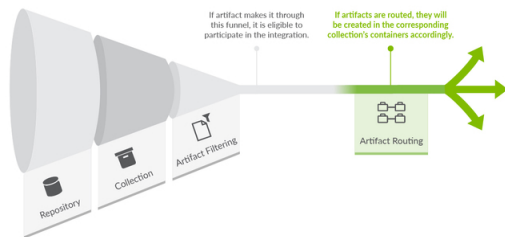
## Next Steps

Once you have completed your Field Flow configuration, your next step will be to review your [Artifact Routing](#).

# Artifact Routing

## Introduction

Once you've configured your [Field Flow](#), your next step will be to configure Artifact Routing.



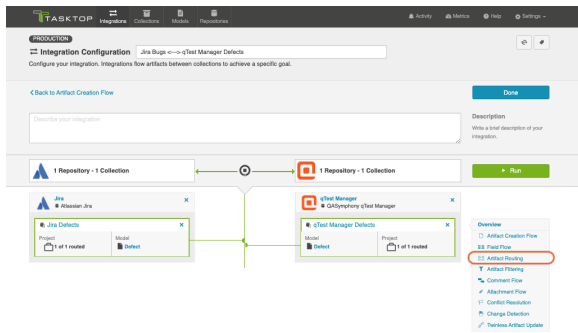
Artifact Routing is needed when artifacts are being created as part of an integration. In addition to knowing the repository in which artifacts should be created, Tasktop also needs to know which container (i.e., project, module, folder, etc) a given artifact should be created in. Specifying the artifact routing does this.

💡 Initially, the artifact routing will determine where an artifact gets created. Over time, if an artifact on either side moves, Tasktop will move the artifact to the corresponding container of the new route, if this is allowed in your repository. If you are moving between lower-level containers, such as sets or folders, this is generally possible. Suppose the move on one side crosses the bounds of the top-level container in the collection. In that case, Tasktop will only attempt to move the artifact if the target repository has writable collection fields. Otherwise, the artifact will remain in the same route, and all non-containment fields will continue to flow as usual.

⚠️ **Note:** If you update the artifact routing on a running integration to include additional lower-level containers, such as sets or folders, please click the ['process all artifacts'](#) button on the Field Flow screen to ensure that all relevant updates are processed.

## Configuring Artifact Routing

To configure Artifact Routing, click the **Artifact Routing** link on the right pane of the **Integration Configuration** screen.

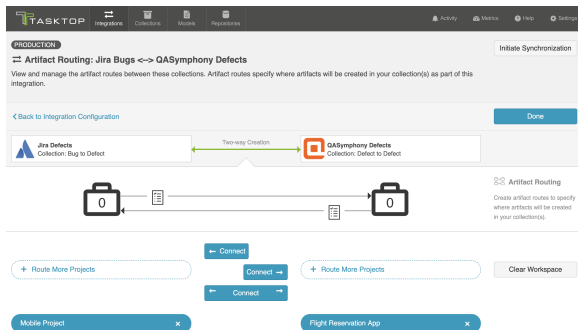


## Static Artifact Routing

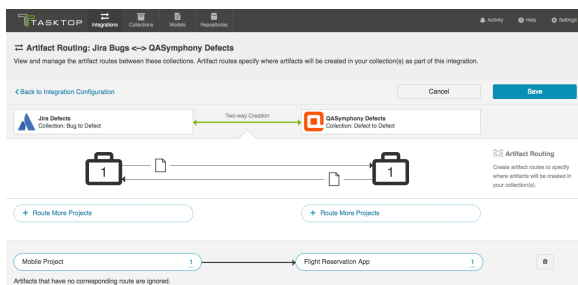
In some cases, the project an artifact resides in the source collection can sufficiently determine which project an artifact should be created in the target collection. In these instances, you can configure what is known as 'static artifact routing' (also known as 'explicit artifact routing').

Static artifact routes can have one or more source projects, but only a single target project.

To configure a static artifact route, use the **Route More Projects** buttons to add projects from your collections to your working space and connect them using the **Connect** button. The directionality on the connect button refers to artifact creation.



In the example shown below, artifacts from the Jira Mobile Project will be created in the Flight Reservation App project in QASymphony.



## Conditional Artifact Routing

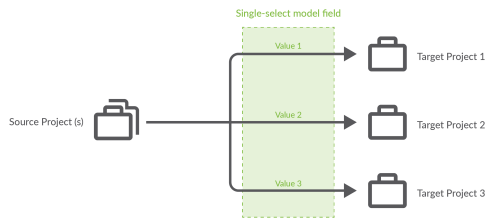
Check out the video below to learn more about Conditional Artifact Routing:

**⚠ Note:** *The video above demonstrates Conditional Artifact Routing within the context of a Create via Gateway Integration. Create via Gateway Integrations are only available in editions that contain the Gateway add-on. See [Tasktop Editions table](#) to determine if your edition contains this functionality. Though the video is for a Gateway Integration, the core concepts outlined in the video can be applied to any integration template.*

In some cases, the project an artifact is in within the source repository does not provide enough information to determine which project the artifact should be created in within its target repository. Oftentimes, in fact, some unique characteristic of an artifact, such as a specific field value, is the factor that should be used to determine which project an artifact should be created in within the target repository.

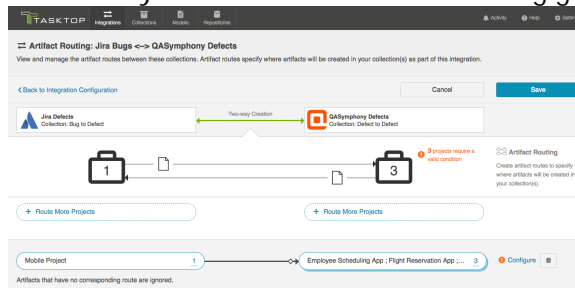
In these instances, you will configure what is known as **conditional artifact routing** to determine which project each artifact is created in within your target repository. Conditional artifact routing (also known as 'dynamic artifact routing') can be used to inspect a single-select field of an artifact and, depending on its value, to route that artifact to the appropriate project in the target collection.

Conditional artifact routes can have one or more source projects, and always have multiple target projects.



To create a conditional artifact route, use the **Route More Projects** buttons to add projects from your collections to your workspace and connect them using the **Connect** button.

Notice that after you've created your conditional artifact routing group, you'll be prompted to



configure your route.

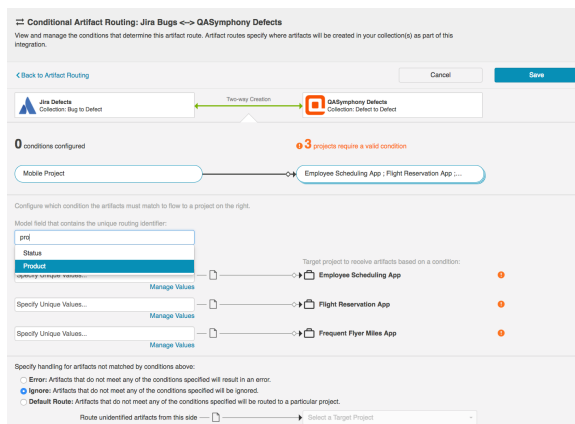
Click **Save** and then click **Configure**.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.

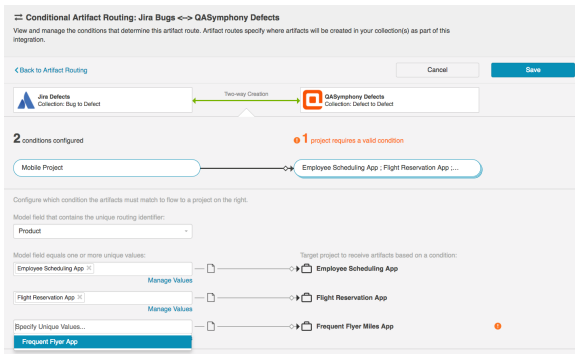
You'll be brought to the Conditional Artifact Routing screen. Here you'll start by selecting the model field that you would like to use to determine your artifact route.

**Note:** Conditional Artifact Routes can only be configured based on **single-select fields** in your model.

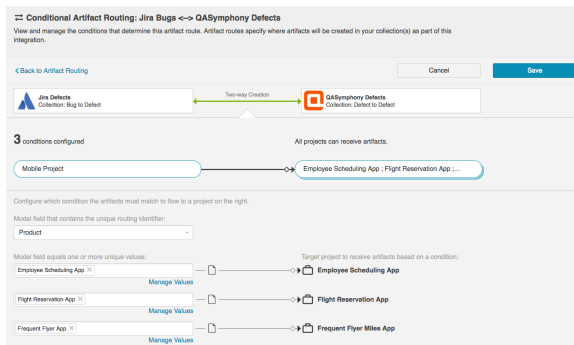
In the example below, the field "Product" contains the unique values that should determine the project an artifact will be created in QASymphony.



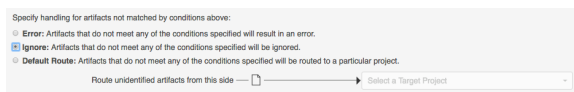
After you select the model field, you can identify one or more value to correspond to each target project. You can also use the **Manage Values** link to select from a list of values.



Once you've done this, you'll see your full conditional artifact routing group:



You can specify how you'd like to handle artifacts that do not meet any of the conditions specified by selecting one of the options provided at the bottom of the screen:



## Next Steps

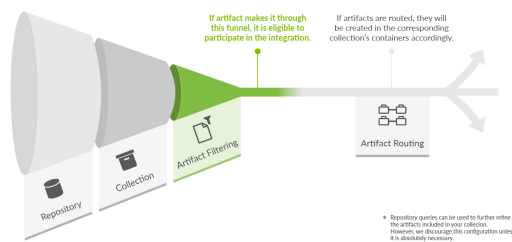
Once you've configured your Artifact Routing configuration, your next step will be to review your [Artifact Filtering](#).

# Artifact Filtering

## Introduction

Once you have completed your [Artifact Routing](#) configuration, your next step will be to review and configure Artifact Filtering.

When configuring your integration, you have several options available to refine which artifacts are eligible to flow. The final mechanism available is *artifact filtering*, which is configured at the Integration level.



**Artifact Filtering** enables you to set filters on an integration in order to limit which artifacts are eligible to flow in your integration.

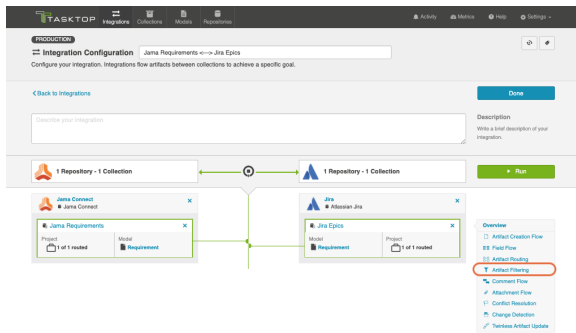
To use a field for artifact filtering, it must:

- Be a part of your model, and be mapped to the collection you are filtering from
- Be one of the following field types:
  - Single Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **only** field values that are already a part of the model will be considered for artifact filtering
  - Multi-Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **only** field values that are already a part of the model will be considered for artifact filtering
  - Boolean
  - Date
  - Date/Time
  - Duration
  - String
  - Long
  - Double

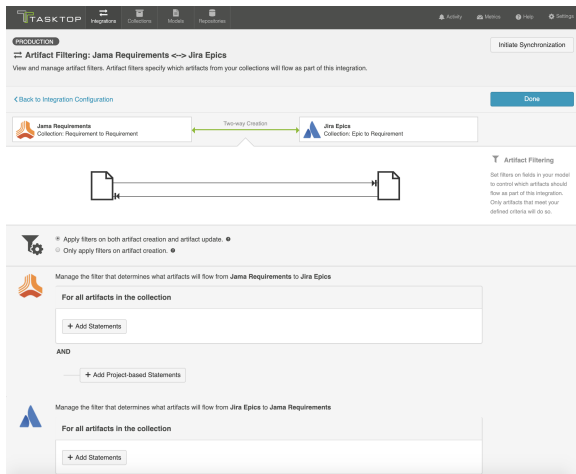
💡 Note that you can utilize our transforms to filter based on an 'unsupported' collection field type, if that field is mapped to a supported field type in your model. For example, you could filter based on a rich text field in your repository, if that rich text field is mapped to a string field in your model.

## Configuring Artifact Filtering

To configure Artifact Filtering, click the **Artifact Filtering** link on the **Integration Configuration** screen.



This will lead you to the Artifact Filtering screen, where you can configure your artifact filters.



## Artifact Creation vs. Artifact Update

First, determine whether you'd like your filter to apply to **artifact creation and artifact update**, or only to **artifact creation**.

*Artifact creation and artifact update* will provide improved performance, and should be used if you don't expect the value for the field you are filtering by to change in the external repositories.

*Artifact creation* means that once artifacts are synchronized, updates will continue to flow between them, even if values are changed that make it such that they no longer meet your filtering criteria. This ensures that your source and target artifacts will stay in sync with one another, even if the value for the field you are filtering by changes.





- Only apply filters on artifact creation. ⓘ
- Apply filters on both artifact creation and artifact update. ⓘ

## Configure Artifact Filter Statements

Next, you can begin configuring your artifact filtering statements. You can add statements for **all artifacts in both collections**, **all artifacts in one collection**, or to **artifacts in specific projects within your collection**.

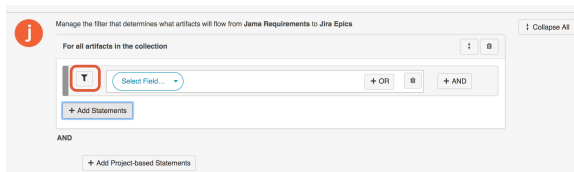
**Note:** If you update your previously saved artifact filter criteria to broaden the scope of your running integration, click the ['process all artifacts'](#) button on the Field Flow screen to ensure that all relevant updates are processed.

## Apply Filter to All Artifacts in Both Collections

To apply a filter to all artifacts in both collections, click **+Add Statements** for all artifacts in your first collection.



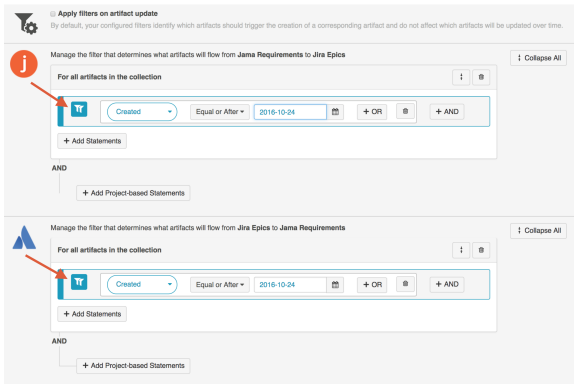
Then, click the filter button. This will apply your filter to the other collection participating in your integration.



You will notice that the button changes to show two filters, indicating that your filter will apply to both collections.

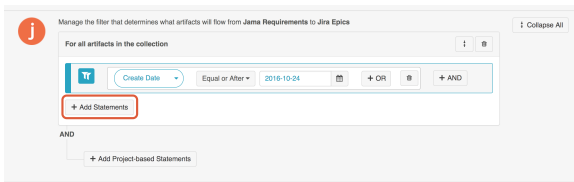
You'll also notice that any modifications you make to that filter statement will automatically be reflected in the other collection. If you'd like to disconnect the filter from both collections, simply click the double-filter button again, and you will be able to edit each filter individually.

Here we are filtering both collections to only create target artifacts that were created on or after October 24th, 2016.

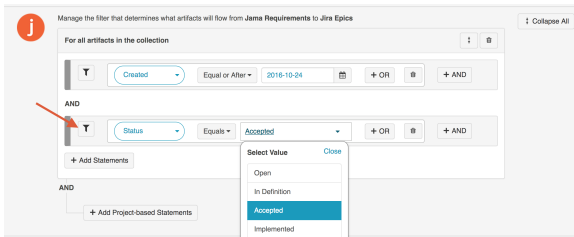


## Apply Filter to All Artifacts in One Collection

To apply a filter to all artifacts in one collection, simply click the **+ Add Statements** button in the desired collection.

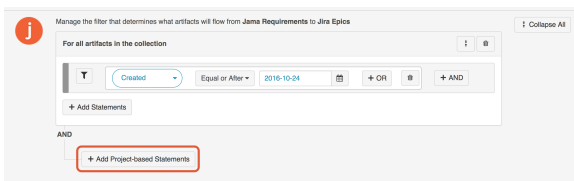


Select your artifact filtering fields and values. You'll see that there is only one filter displayed on the left, which tells you that this filter only applies to one collection in your integration.

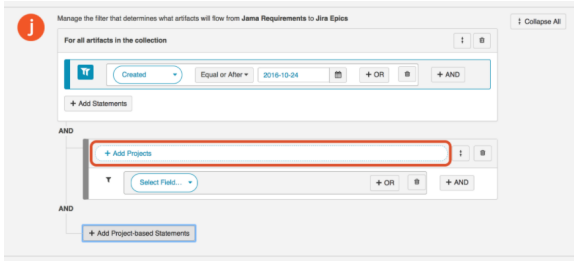


## Apply Filter to Artifacts within Certain Projects in a Collection

To apply a filter to artifacts within certain projects in a collection, click **+ Add Project Based Statements**

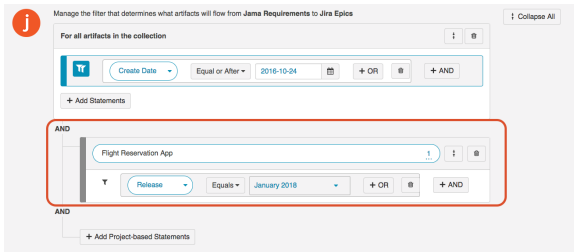


Click **+ Add Projects** to select your project.



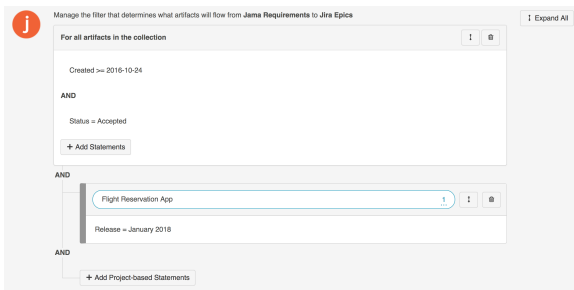
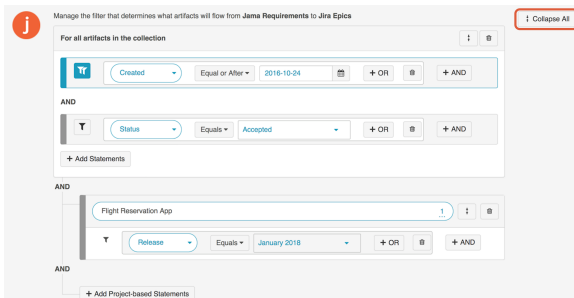
Select the project(s) you'd like your filter to apply to.

Then click **Select Field...** to begin configuring your filtering statement.



## Viewing Artifact Filter Statements

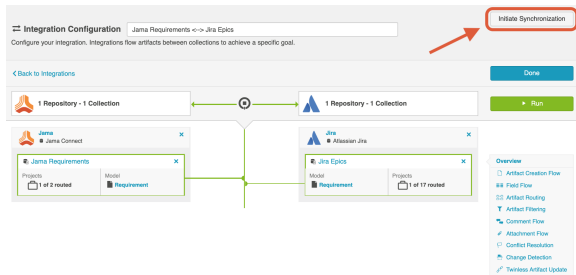
You can click the **Collapse All** button to view an easy-to-read version of your artifact filtering statements.



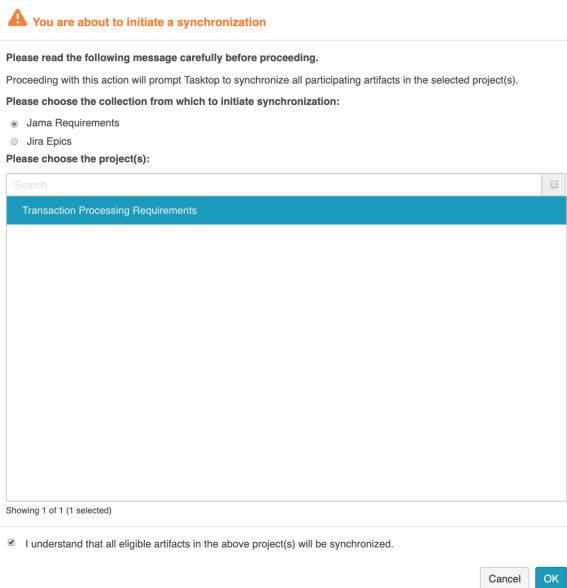
## Expanding Artifact Filters

If you update an Artifact Filter of a running integration so that it includes additional artifacts, you can choose to initiate a synchronization immediately in order to synchronize the newly eligible artifacts.

To initiate synchronization, go to the main integration configuration screen and click **Initiate Synchronization** in the upper right corner.



On the pop-up that appears, select the collection and project(s) whose artifacts you'd like to synchronize:



This will immediately trigger a special high fidelity full scan for the project(s) selected, causing eligible artifacts in those project(s) to synchronize.

## Next Steps

Once Artifact Filtering is configured, your next step will be to review and configure [Comment Flow](#).

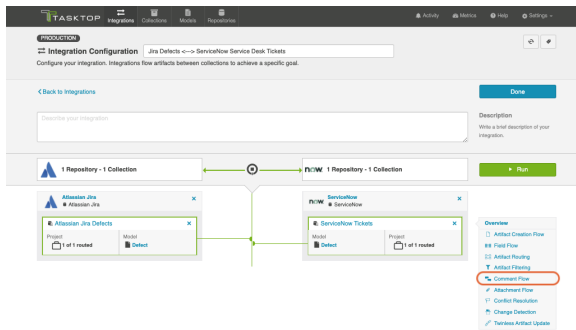
# Comment Flow

## Introduction

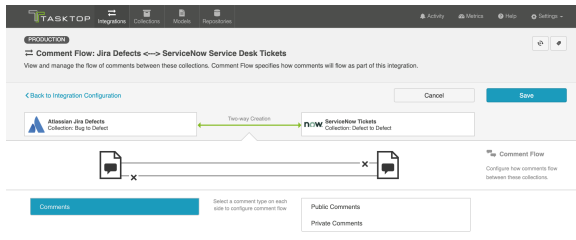
Once you've configured [Artifact Filtering](#), your next step will be to review and update Comment Flow.

## Configuring Comment Flow

To enable and configure Comment Flow, click the **Comment Flow** link on the **Integration Configuration** screen.

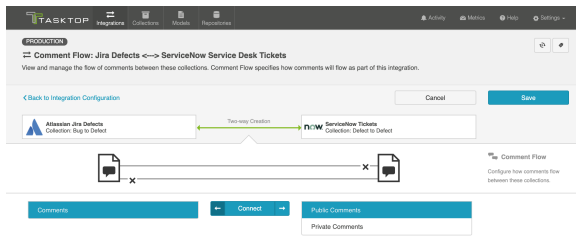


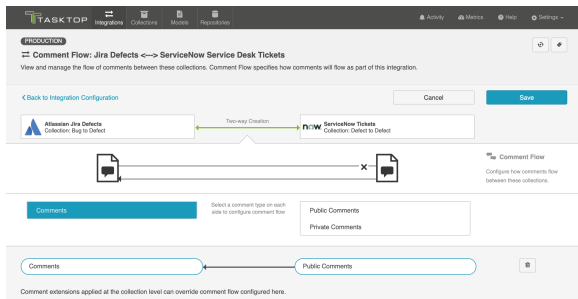
This will bring you to the Comment Flow screen.



If your collections support comment flow, you will be able to choose your desired Comment Flow style here. You can choose to flow comments bi-directionally or in a single direction.

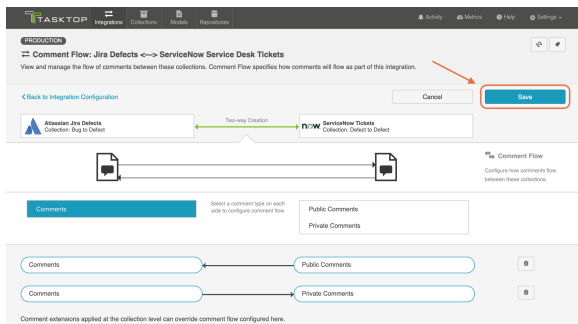
Comment Flow can also be fine-tuned based on comment privacy levels if one or more repositories in the integration supports the notion of public vs. private comments.





Once you've updated the comment flow settings as desired, click **Save** and **Done** to save your changes.

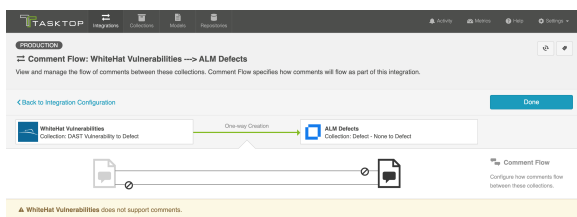
**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.



Below are some details to be aware of when flowing comments:

- Tasktop will **not** process **deletions** of comments that have already flowed.
- When comments that have already flowed are **edited** in the source repository, a new comment will be created in the target repository containing the updated text. The original comment in the target repository will be unchanged.
- **Synthetic Comments** (auto-generated comments made when a new attachment is added, a status of an artifact changes, etc.) are not supported by Tasktop.

You can check our [Connector docs](#) to see which repositories support comment flow. If one or both of your collections does not support comment flow, you will see a notice like the one below:



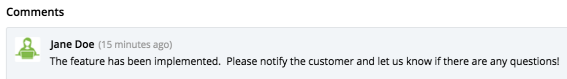
## Comment Impersonation

Comment Impersonation refers to Tasktop's ability to assign a specific user to a given comment. You can learn if your repository supports impersonation by viewing our [Connector docs](#).

Depending on whether or not impersonation is supported, your comments may flow over to your target repository in one of two ways:

- When your target repository supports impersonation, Tasktop will assign the comment to the proper user if it is possible to locate the user with the information provided on the source artifact.

In cases like this, your comment will appear as though it were created by the corresponding user, as seen in the comment below:



On the other hand,

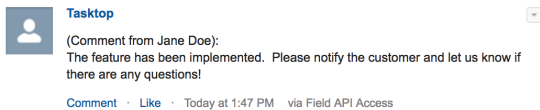
- When your target repository supports impersonation, but Tasktop cannot locate the person with the information provided from the artifact in the source repository,

Or

- When your target repository does not support impersonation,

The comment will appear in your target repository as though it were created by the default user associated with your repository configuration in Tasktop, and the name of the user who truly recorded the comment will be listed at the beginning of the comment text.

In cases like the final two outlined above, your comment will look like this:



## Next Steps

Once you've completed your Comment Flow configuration, your next step will be to review and update your [Attachment Flow](#).

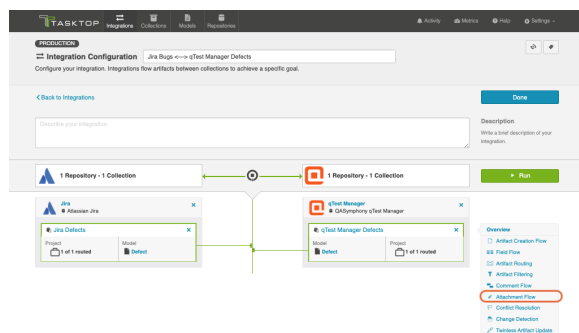
# Attachment Flow

## Introduction

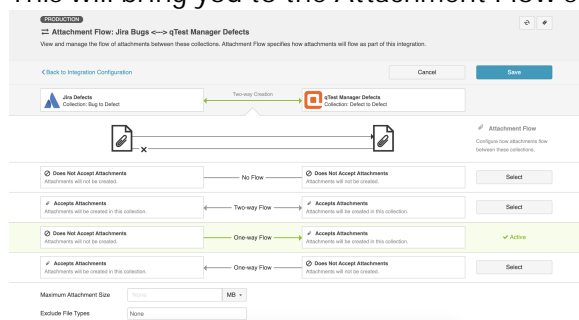
Once you've configured [Comment Flow](#), your next step will be to configure Attachment Flow.

## Configuring Attachment Flow

To enable and configure Attachment Flow, click the **Attachment Flow** link on the **Integration Configuration** screen.



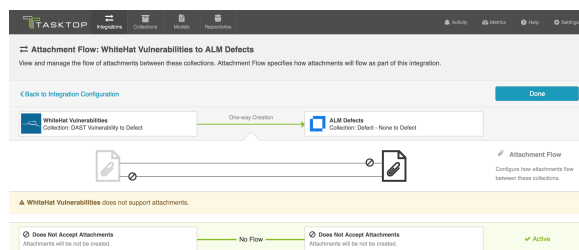
This will bring you to the Attachment Flow screen.



If your collections support attachment flow, you will be able to choose your desired Attachment Flow style here. You can choose to flow attachments bi-directionally or in a single direction.

**Note:** Attachment Flow only flows new attachments. Deletions of existing attachments will not flow.

You can check our [Connector docs](#) to see which repositories support attachment flow. If one or both of your collections does not support attachment flow, you will see a notice like the one below:

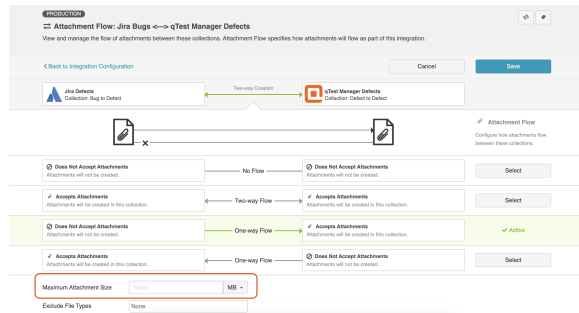




# Maximum Attachment Size

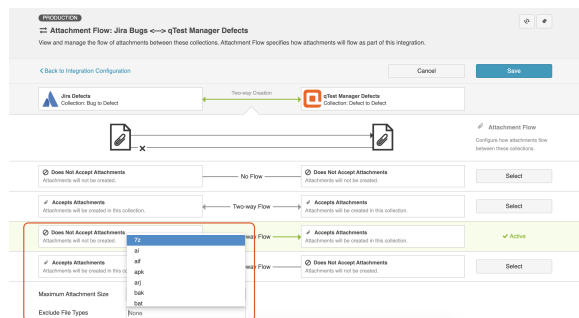
You can also configure the **maximum attachment size**. If attachments are larger than this size, they will be ignored by your integration.

💡 If you are unsure of the maximum attachment size allowed in your repository or if you leave this field blank and it turns out that the attachment is, in fact, larger than the maximum size the repository allows, you will see an error message in Tasktop for that attachment. You can then deduce, based on the error message in Tasktop, what the maximum size is, and use that data to populate the field on the Attachment Flow screen.



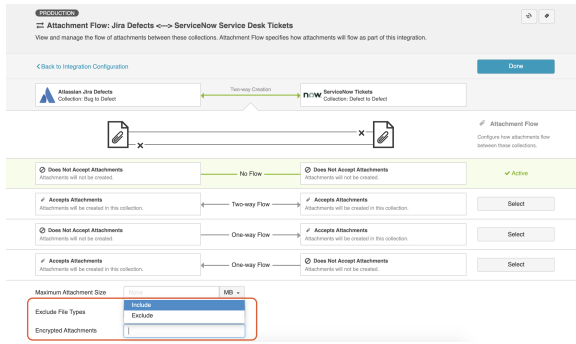
# Exclude File Types

If you'd like to exclude certain file types from your attachment flow, select the file type(s) from the dropdown menu here.



# Encrypted Attachments

If one of your repositories (such as ServiceNow) supports encrypted attachments, you will see an option to include or exclude encrypted attachments from attachment flow.



## Attachment Impersonation

Attachment Impersonation refers to Tasktop's ability to assign a specific user to a given attachment. You can learn if your Desk Tickets repository supports impersonation by viewing our [Connector docs](#).

Depending on whether or not impersonation is supported, your attachments may flow over to your target repository in one of two ways:

- When your target repository supports impersonation, Tasktop will assign the attachment to the proper user if it is possible to locate the user with the information provided on the source artifact.

On the other hand,

- When your target repository supports impersonation, but Tasktop cannot locate the person with the information provided from the artifact in the source repository,
- Or,
- When your target repository does not support impersonation,

The attachment will appear in your target repository as though it were created by the default user associated with your repository configuration in Tasktop.

## Next Steps

Once you've completed your Attachment Flow configuration, your next step will be to review and update your [Conflict Resolution](#).

# Conflict Resolution

## Introduction

Once you've configured your [Attachment Flow](#) , your next step will be to review and update your Conflict Resolution Strategy.

When two-way field flow is configured, data conflicts become possible. A data conflict will occur if a field on an artifact is modified on both the source artifact and target artifact during the same [Change Detection Interval](#). The Change Detection Interval refers to how often Tasktop checks repositories for changes to artifacts.

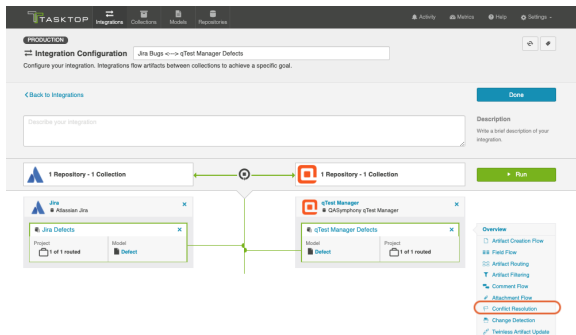
The Conflict Resolution Strategy screen allows you to control how data conflicts will be resolved:

1. **Error Upon Conflict:** An error will be generated, and no updates will be made to the conflicted field, or to any other fields on the artifact. The error message will notify you that the conflict occurred and will provide steps on how to resolve the conflict. *Note that once a conflict is detected, no subsequent updates will be made to the artifact pair until the conflict is resolved.*
2. **Left Collection Dominates:** Values from the artifact in the left collection will overwrite the values in the right collection.
3. **Right Collection Dominates:** Values from the artifact in the right collection will overwrite the values in the left collection.

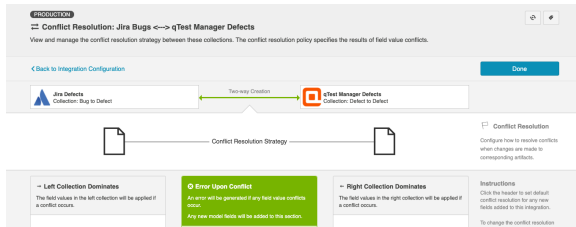
## Configuring Conflict Resolution

To prevent data conflicts, you will need to configure your conflict resolution strategy. This will ensure consistency across your source artifacts and target artifacts if field values of the same artifact are modified.

To select your Conflict Resolution Strategy, click the **Conflict Resolution Strategy** link on the right side of the Integration configuration screen:



This will lead you to the Conflict Resolution Screen, where you can select your desired policy:



Once selected, click **Save** and **Done**. This will bring you back to the Integration Configuration screen.

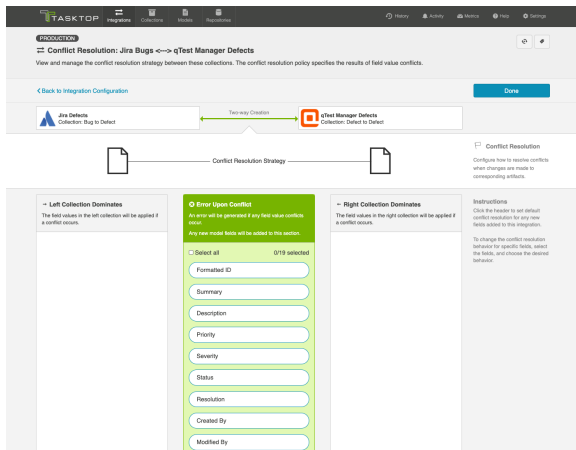
## Field-by-Field Conflict Resolution

See [Tasktop Editions table](#) to determine if your edition supports **Field-by-Field Conflict Resolution functionality**.

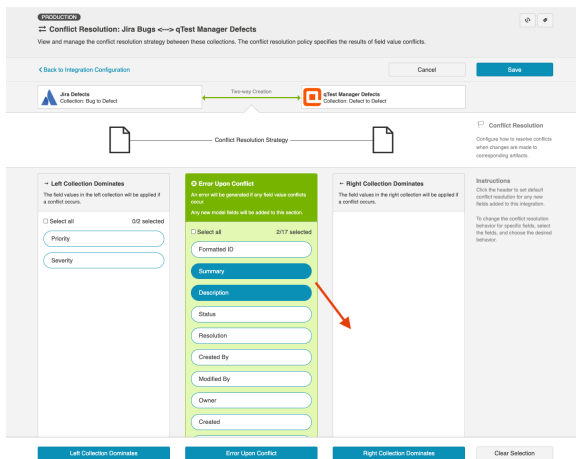
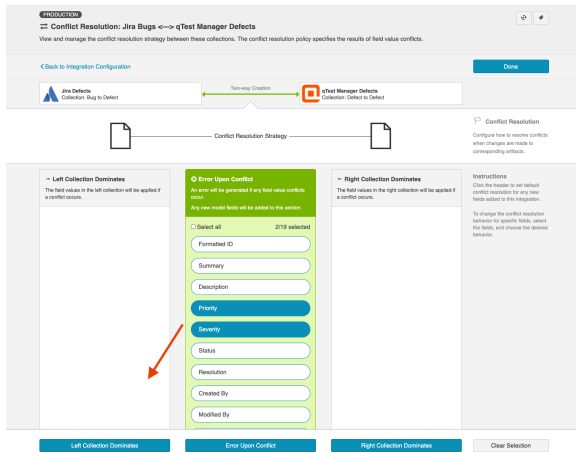
You can also configure your conflict resolution strategy on a field-by-field basis.

To specify field-by-field conflict resolution, you will follow the same steps listed above; however, instead of selecting a single policy for all values, you can choose individual field values and determine the conflict resolution policy for each value. If you would like to set a default behavior for new field values, you can do so by clicking the header of each column.

**Note:** Any new fields added will default to 'Error Upon Conflict' unless the default behavior is changed.



As you can see in the scenario below, the user has chosen their Jira collection to dominate for the field values **Priority** and **Status** and their qTest collection to dominate for the field values **Summary** and **Description**.

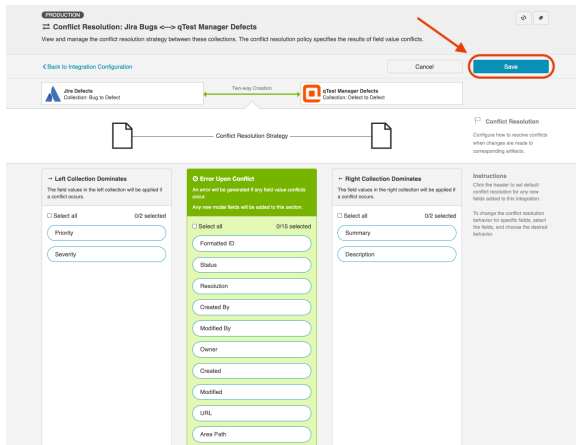


If any of these field values are updated within the same change detection interval (e.g., **Priority** is set to **High** in Jira and **Low** in qTest), the dominating collection will overwrite the values in the non-dominating collection (i.e., **Priority** will be updated to **High** in qTest).

**Note:** If a conflict occurs for a field whose policy is set to **Error Upon Conflict**, all fields will be blocked from being updated on the artifact pair, until the error is resolved. This includes fields whose policy is set to 'Left/Right Collection Dominates'

After you have configured your desired conflict resolution policy, click **Save** and **Done** to save your changes.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.



## Next Steps


Once you've selected your Conflict Resolution Strategy, your next step will be to review and update your [Change Detection settings](#) for this integration.

# Change Detection

## Introduction

Once you've configured your [Conflict Resolution](#), your next step will be to configure your Change Detection settings.

Tasktop's default global change detection settings can be found on the [General \(Settings\)](#) screen. If you'd like to override the global defaults, you can configure integration-specific change detection and full scan intervals on this screen.


 Note that you can also set repository-specific change detection on the [Repository Connection](#) screen.

- The **Change Detection Interval** is the time between polling requests to detect *only changed artifacts*. This defaults to 1 minute on the General (Settings) screen, but can be customized as desired.
- The **Full Scan Interval** is the time between polling requests to detect changed artifacts, in which all artifacts that have previously synchronized in the integration are scanned.

**Note:** For **Hub Cloud instances**, we recommend setting the change detection interval to at least 1 minute and the full scan interval to at least 24 hours.

Not all changes to an artifact will register as a change. Some repositories do not mark items as changed when (for example) a relationship is added or an attachment has changed. These may not be picked up by regular Change Detection, but will be picked up by a Full Scan. You can review our [connector docs](#) to see types of updates that will require a full scan. The Full Scan Interval defaults to 24 hours on the [General \(Settings\)](#) screen, but can be customized as desired.

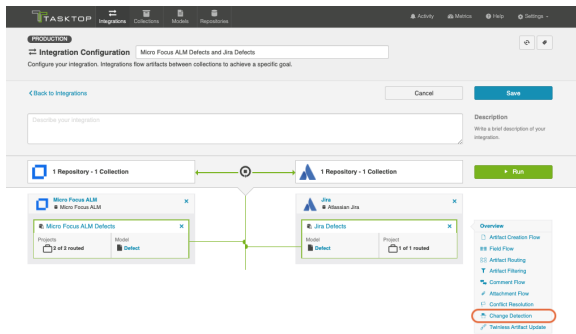
Note that since the Full Scan only scans artifacts that have previously synchronized, artifacts that are newly eligible for synchronization due to updated artifact filtering or routing will not be picked up by the Full Scan. These artifacts will only be processed by clicking the '[process all artifacts](#)' button, or when a new integration-eligible change is made to them.

 When configuring change detection settings, integration-level change detection has the highest precedence, followed by repository-level change detection, and then global change detection. This means that if integration-level change detection is configured, the integration-level setting will **always** be used (even if repository-level or global settings are configured). If no overrides are set, change detection will default to the global settings.

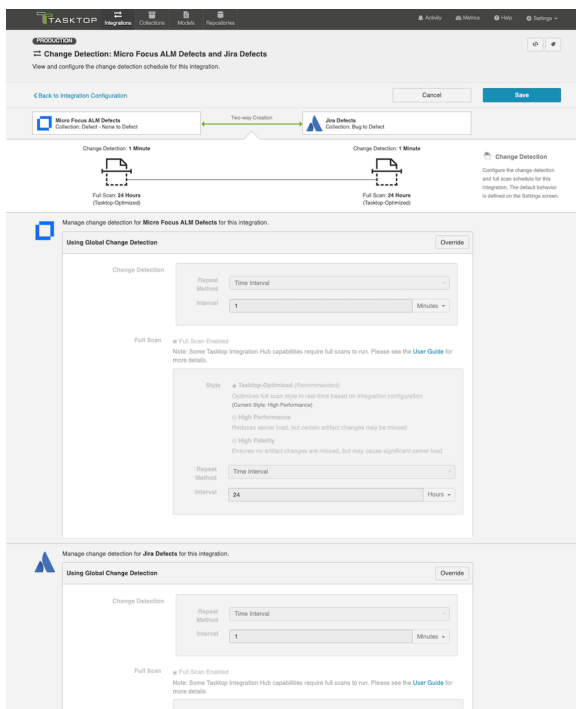
You can learn more about change detection and full scan styles in our FAQ [here](#).

## Configuring Change Detection

To configure integration-specific change detection, click the **Change Detection** link on the **Integration Configuration** screen.



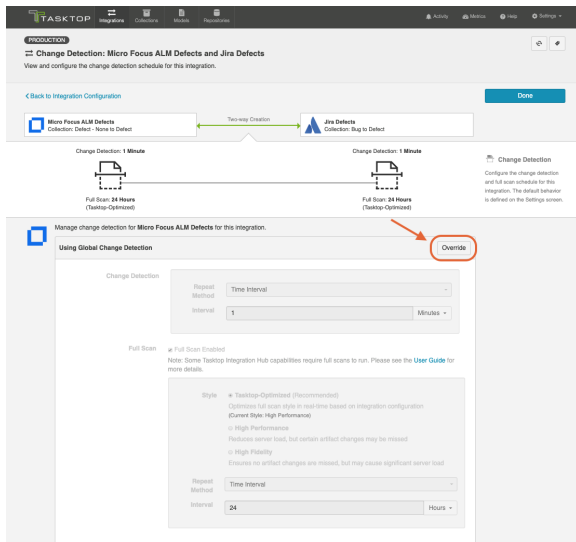
This will bring you to the Change Detection screen, where you can view the current change detection and full scan intervals configured for each collection in this integration. These will default to the global intervals configured on the General (Settings) screen.



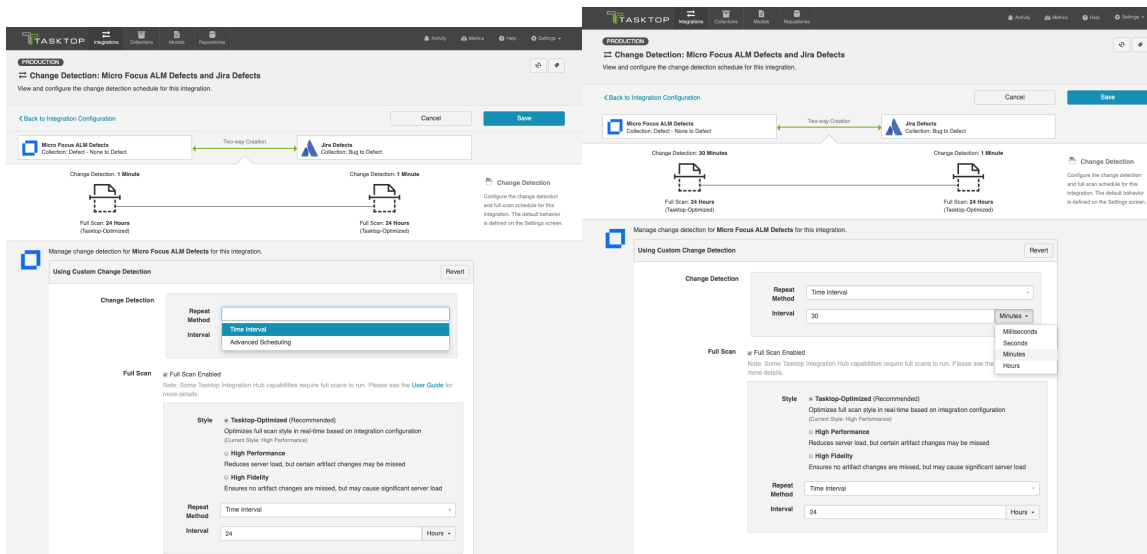
To override the current settings, click the **Override** button on your desired collection. This will allow you to set a custom change detection and/or full scan interval for each collection within the context of this integration.

💡 Note that these custom settings will only impact *this* integration; they will not impact other integrations that use the same collections.





Once you click **Override**, you will be able to configure custom change detection and full scan intervals for that collection within the context of the integration:



## Full Scan Style and Interval

In addition to customizing the full scan interval, you can also select your desired full scan style in order to best meet your specific performance and server load needs.

The following full scan styles are available:

- **Tasktop-Optimized (Recommended):** This is the default selection. It optimizes your full scan style in real-time based on your integration configuration.
- **High Performance:** This full scan style reduces server load, but certain artifact changes may be missed.
- **High Fidelity:** This full scan style ensures no artifact changes are missed, but may cause significant server load.

If High Performance style is selected, Tasktop will provide a warning identifying any specific artifact changes which may be missed:

 You are about to apply High Performance Full Scan to Jira Defects in this integration

Please read the following message carefully before proceeding.

The following 1 configured fields in Jira Defects require the High Fidelity Full Scan style to detect changes. The High Performance Full Scan style may cause changes to these fields to be missed.

- Watchers (watches)

The following 4 unconfigured fields in Jira Defects require the High Fidelity Full Scan style to detect changes. Please be aware of this when modifying your configuration.

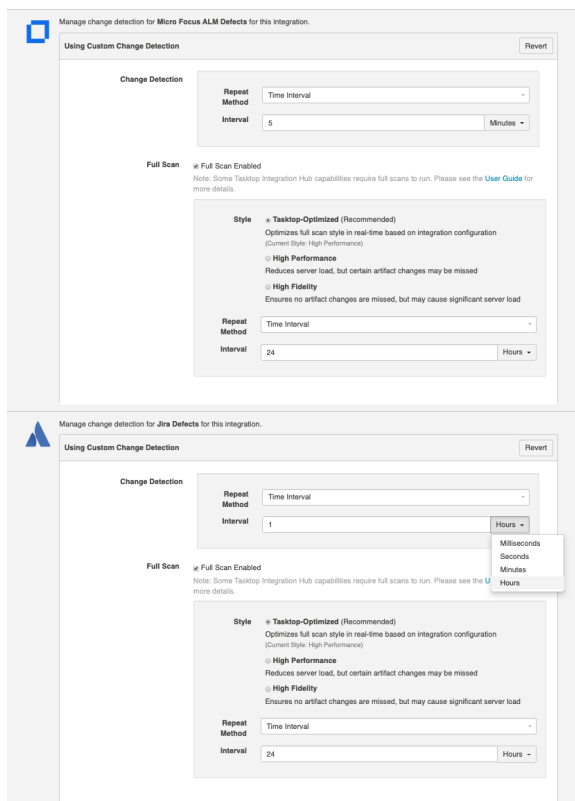
- Time Spent (timespent)
- Remaining Estimate (timeestimate)
- Original Estimate (timeoriginalestimate)
- Web Links (web-links)

[Show Fewer](#)

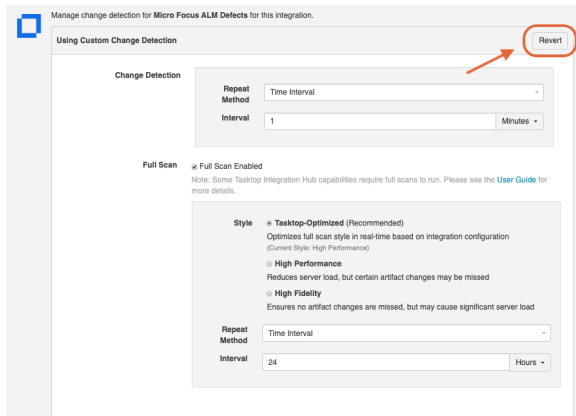
Are you sure that you would like to apply the High Performance Full Scan style?

- I understand that applying the High Performance Full Scan style may cause some artifact changes to be missed.

Once you have configured your change detection and full scan intervals for your first collection, you can update the settings for the remaining collection, if desired.

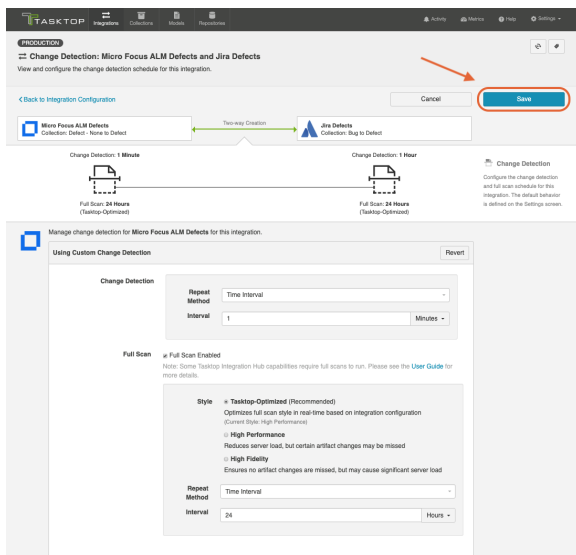


If you'd like to restore the global change detection settings, simply click the **Revert** button to remove the custom settings:



Once you've updated the change detection settings as desired, click **Save** and **Done** to save your changes.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.

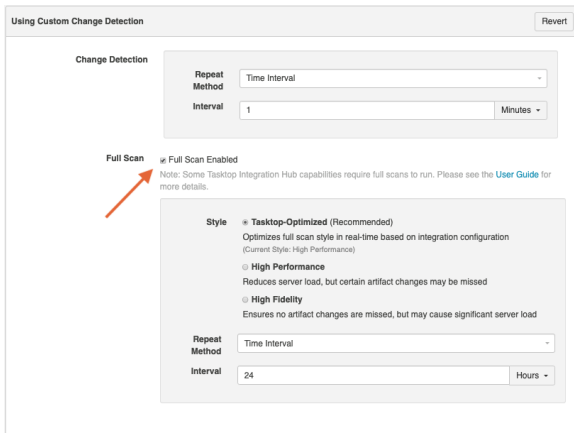


## Disabling Full Scan

Full scans can be disabled globally on the [General \(Settings\)](#) screen, or on a per-integration basis via the Change Detection screen. This feature is especially useful for users that do not want to overload their repositories.

**Note:** Disabling full scans does not disable manually requested full scans. Learn more about manually requested full scans in our FAQ [here](#).

To disable full scan, uncheck the **Full Scan Enabled** box for the desired collection.



⚠ If you choose to disable full scans, note that twinless artifact updates will not work and some artifact updates may be missed.

## Configuring Change Detection with Cron Expression

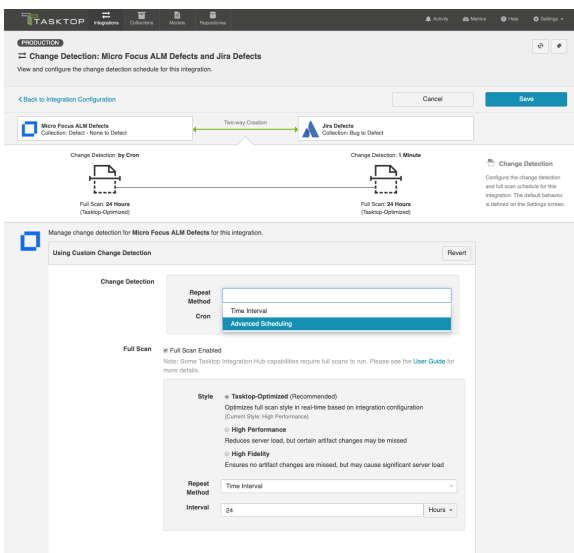
*Based on your edition, you may have the ability to configure change detection using a Cron Expression. To do so, please see the section below.*

You can use cron expression to schedule change detection or full scan to your desired intervals. For example, you may only want to perform a full scan Monday through Friday during work hours, to lighten the load on your repositories, and to receive updated information during your workday. Using cron expression, you can configure such complex schedules by running change detection or full scan during certain hours of the day, certain days of the week, or specific days of the month.

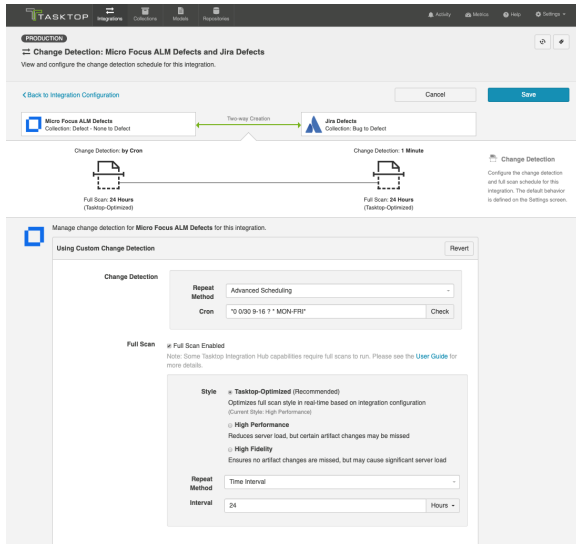
💡 **Note:** Cron expressions must be written using the Quartz cron format to be valid.

Learn more in our FAQ [here](#).

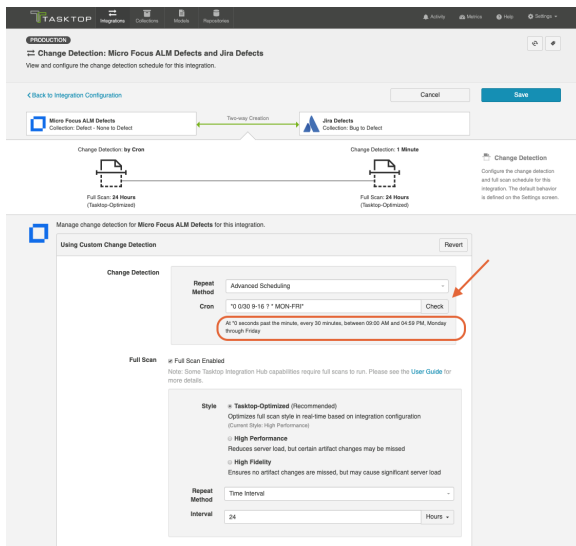
To utilize cron expression for your full scan or change detection, select **Advanced Scheduling** in the Repeat Method field.

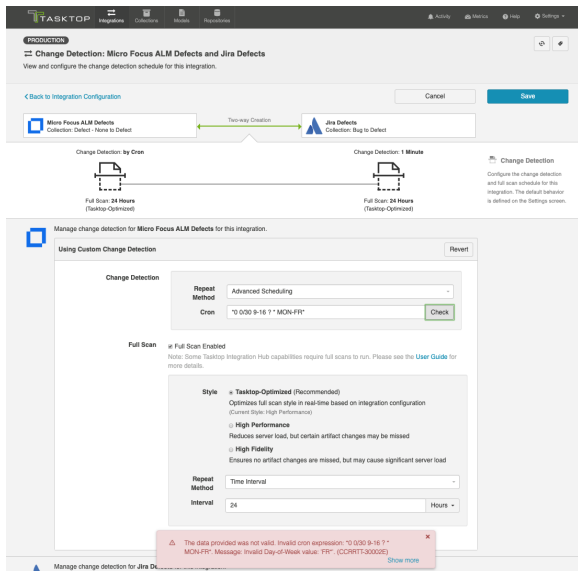


You can then enter the cron expression for your desired scheduling. For example, if you would like to run a full scan every 30 minutes from 9am to 5pm from Monday through Friday, it would be written as follows: `*0 0/30 9-16 ? * MON-FRI*`



You can check to see if your cron expression is valid by clicking the **Check** button. If valid, a readable form of the cron expression will be displayed. If the cron expression is invalid, an error message will appear.





## Next Steps

Once you've selected your Change Detection settings, your next step will be to configure your [Twinless Artifact Update](#) settings for this integration.

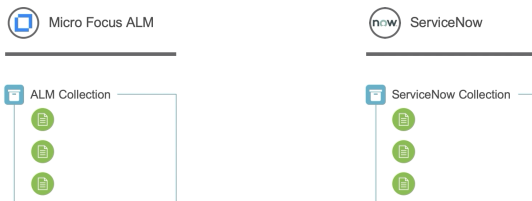
# Twinless Artifact Update

See [Tasktop Editions table](#) to determine if your edition contains Twinless Artifact Update functionality.

## Introduction

Once you've configured your [Change Detection](#) settings, your next step will be to configure your Twinless Artifact Update settings.

The **Twinless Artifact Update** screen allows you to configure one final update (for example a comment or a status change) on an artifact when its "twin" in the other repository is no longer eligible to participate in the integration (e.g., when it's been deleted or no longer meets the artifact filter). The final update informs the newly twinless artifact that the synchronization has been discontinued.



This feature demystifies the integration process and allows end users to understand why an artifact may no longer be receiving updates via the Tasktop integration. Once notified of the change via a comment or field update on the artifact, users can work with their Tasktop admin or with users in the other system to troubleshoot.

Artifacts are no longer eligible to participate in an integration if one or both of the conditions below are met for an endpoint:

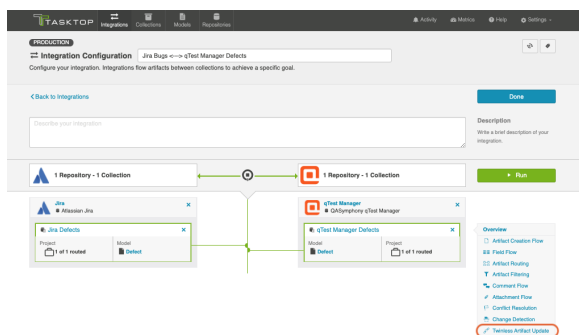
1. Artifacts fall out of [Artifact Routing](#). This can happen when:
  - a. Artifacts are deleted
  - b. Artifacts are moved to projects not routed as part of the integration
2. Artifacts no longer meet the [Artifact Filtering](#) criteria.

**Note:** If an artifact leaves the integration due to a [repository query](#) and twinless artifact update is configured with the "fall out of Artifact Routing" option, twinless artifact update may not trigger. [Contact Tasktop Support](#) for more details.

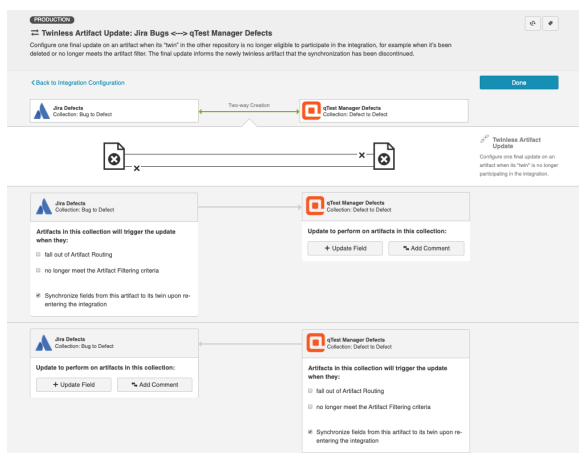
You can configure the update to occur based on one or both of the criteria above, as well as the type of update made to the 'twin' artifact. For example, you could add a comment saying "Artifact is no longer synchronizing" or change the state of the twin to "No Longer Synchronizing" or clear out a field value.

## Configuring Twinless Artifact Update

To configure your Twinless Artifact Update settings, click **Twinless Artifact Update** on the **Integration Configuration** screen.



This will lead you to the Twinless Artifact Update configuration screen.



Select the conditions you'd like to trigger the update, along with the specific update that you'd like to occur on the other side.

Depending on the type of update and the repositories used, it may take until the next High Performance Full Scan for Tasktop to detect that an artifact is no longer eligible for the integration.

If you are **updating a field**,

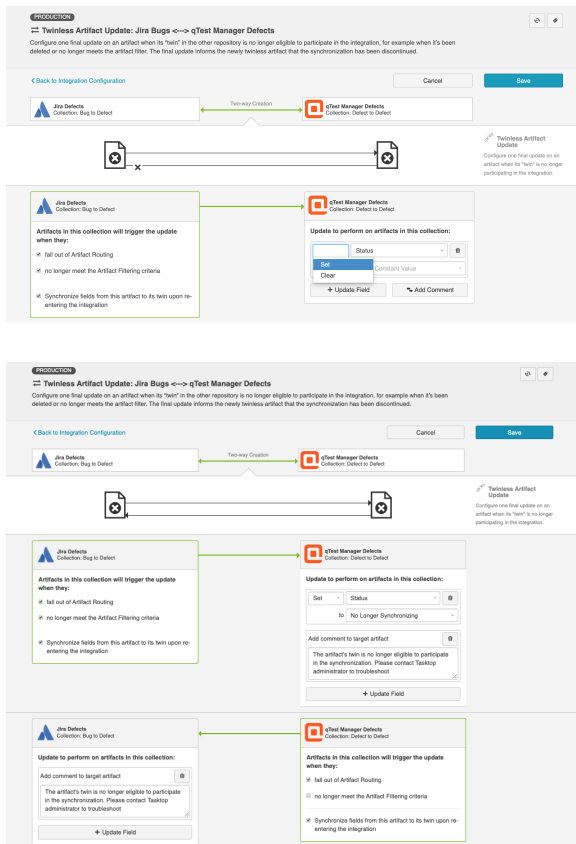
- any model field that can have a **constant value** set will be available
- any model field that can have a field value cleared will be available
- the value you set will **over-write** any existing values in that field
- multiple field updates can be configured for each collection

If you are **adding a comment**,

- the connector for that collection must support comments (review our [Connector Docs](#) to confirm)
- if public and private comments are supported in your collection, you can specify if the comment is private or public
- rich text is not supported
- you do not need to configure **comment flow** for the comment to appear



**Note:** Twinless Artifact Update only flows constant values for model fields configured in the Twinless Artifact Update screen. If a field is updated and the artifact is deleted, Tasktop may not retrieve the artifact in the interim to know the value of the field. When joining fields via extensions, this can cause issues. For example, if you have a field labeled **Field A** and another field labeled **Field B** and they are both flowing into the model and then they are joined and written to a **single** field in a repository; if Twinless Artifact Update is set to provide a constant value for **Field A** and **Field B** isn't available in the model for the extension, the field in the repository cannot be updated. In such cases, it is recommended to provide a value for **Field B** within the Twinless Artifact Update configuration.



By default, your configuration will be set to automatically synchronize fields from each artifact to its twin upon its re-entry to the integration. If you would not like to force an update at that time, you can un-check that box.

Once you've configured your settings, click **Save** at the top of the screen, and then **Done**.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.

## Next Steps

Congratulations! Configuring the Twinless Artifact Update is the final step in configuring your Work Item Synchronization! You are now ready to [run your integration](#).

# Running Your Integration(s)

## Introduction

Once you've completed your [Work Item Synchronization](#) configuration, it's time to run your integration!

## Integration Impacts

**⚠** Please be aware that integrations will trigger changes in your end repositories and that misconfiguration can cause items to be duplicated or modified in unexpected ways. Additionally, there is no 'undo' in Tasktop or the repositories. If you have any questions, please [contact support](#).

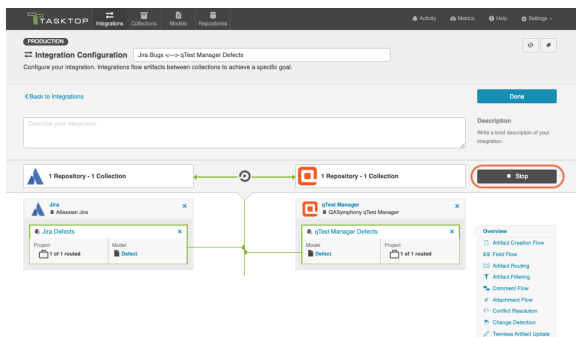
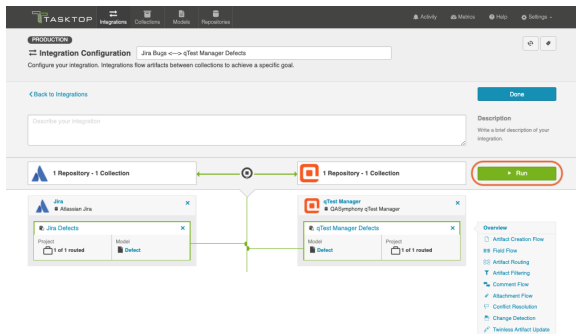
Tasktop does *not* support deletion of artifacts across repositories. See this [FAQ](#) for additional information.

## Running your Integration

There are two ways to start or stop your integration:

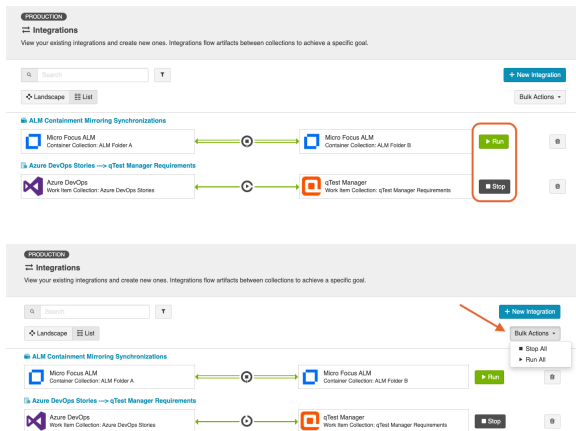
### From the Integration Configuration Screen

Simply click **Run** to run the integration, and **Stop** to stop the integration.



### From the Integrations List Screen

Click **Run** or **Stop** next to each integration you would like to update. You can also use the **Bulk Actions** button to run or stop all integrations.



## Next Steps

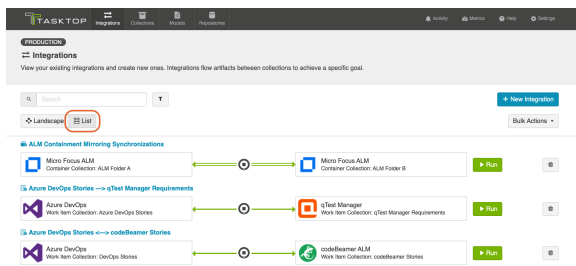
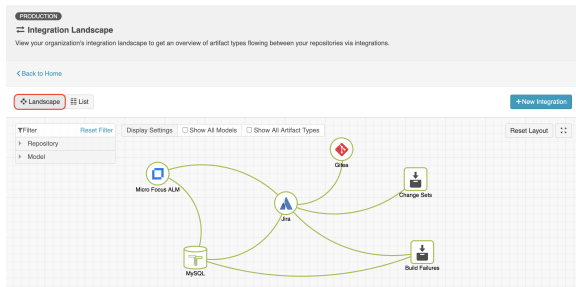
You can learn how to view and visualize your integrations [here](#).

# Viewing Your Integration(s)

## Viewing Your Integrations

See [Tasktop Editions table](#) to determine if your edition contains Integration Landscape View functionality.

When viewing your integrations, you have the option of viewing them in either Landscape or List mode.



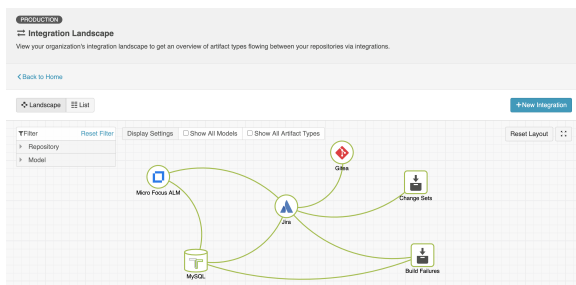
## Landscape View

See [Tasktop Editions table](#) to determine if your edition contains Integration Landscape View functionality.

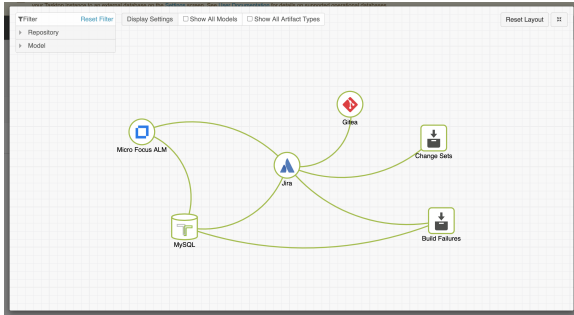
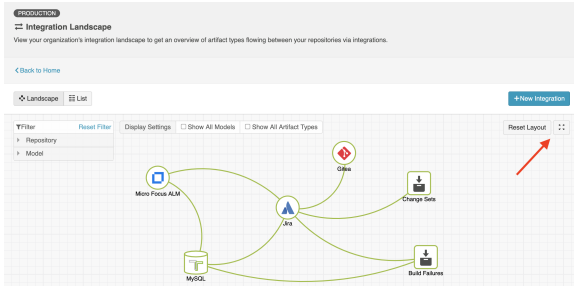
Learn more about the Integration Landscape View in the video below:

Tasktop will default to the Landscape View, which enables you to visualize your entire integration landscape and see how your integrations relate to one another. Use our built-in filters to see as little or as much information as you'd like!

Here's a simplified view:



To see the Landscape View in full screen mode, click here.

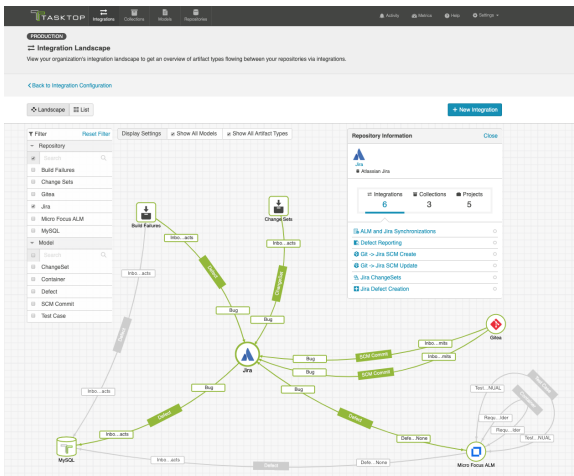


If you'd like to see additional information, you can utilize the filters, or click on a repository node to modify which information is shown.

Some examples of additional information you can see are:

- Models
- Artifact Types
- Artifact Creation Arrows
- List of all relevant integrations (see this by clicking on the repository node)
  - Indicator of whether each integration is running or not
- Collections
- Projects

Here's an example of a more detailed view:

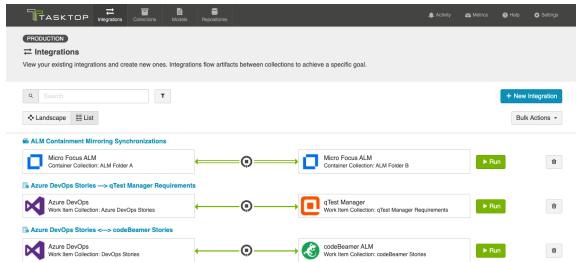


# List View

If you'd like, you can toggle to List View, which will show you a list of all integrations you have created.

You can use this view to:

- Start an Integration
- Stop an Integration
- Delete an Integration
- Click into an Integration and modify its configuration



# Tips and Tricks

The following pages contain information and best practices for common Work Item Synchronization use cases:

- [Synchronizing Relationships](#) : Tasktop affords you the ability to not only flow various artifacts between your collections but also to mirror the relationships between those artifacts. This page will explain how to configure both Internal Artifact Relationship Management (ARM) and External Artifact Relationship Management (ARM). Internal ARM refers to the ability to flow artifacts, along with their internal relationships from your source repository to your target repository. External ARM refers to a more lightweight approach that allows you to flow links to related artifacts in your source repository to a string or weblink field on your target artifact.
- [Synchronizing an Artifact ID or URL Reference](#) : In order to provide traceability, Tasktop affords you the ability to flow the ID or URL for the source artifact to a string or weblink field on the target artifact, thus enabling you to easily navigate between the two. This page explains how to configure that scenario.

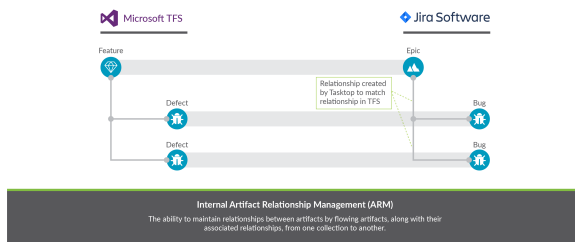
# Synchronizing Relationships

## Synchronizing Relationships

Tasktop affords you the ability to not only flow various artifacts between your collections, but also to mirror the relationships between those artifacts. This is referred to as Artifact Relationship Management (ARM). There are two types of ARM: Internal ARM and External ARM. We will outline both types below.

## Synchronizing Internal Relationships

Below, we'll outline an **Internal ARM** scenario where we flow Microsoft TFS features to Jira epics, in addition to the defects that block them, all while preserving the relationships between the artifacts within each internal system.

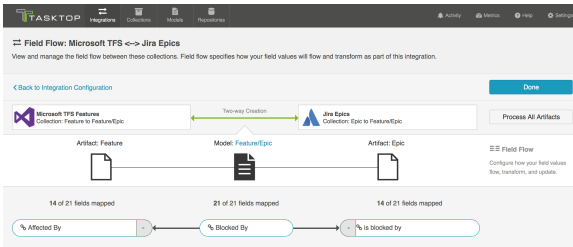


Here's how to configure this scenario in Tasktop:

💡 First, confirm that both repositories support relationships in our [Connector Documentation](#).

1. To flow these artifacts along with their relationships, we will need to configure two integrations (and four collections):
  - a. Microsoft TFS Features Jira Epics, with 'blocked by' relationship field mapping
  - b. Microsoft TFS Defects Jira Defects
2. First, configure your Feature Epics Synchronize Integration
  - a. Ensure that your model includes a 'blocked by' relationships field
    - i. 💡 In general, we recommend using the 'relationships' field type in your model, rather than 'relationship,' especially in cases where you may want to map a 'relationship' field in one repository to a 'relationships' field in your other repository.
  - b. On each Collection, click 'configure relationship types,' and map the 'blocked by' model field to the appropriate relationship field ('affected by' in TFS and 'is blocked by' in Jira).
  - c. On your Integration Field Flow page, you will see the two relationship types mapped to one another.





3. Next, configure your Defect Defect Synchronize Integration as you normally would.
4. Run both integrations. You will see your epics and features, and your defects, as well as *their relationships to one another* successfully flow as part of your integration.

**Note:** If you are configuring an integration between different collections of the same repository (i.e., to flow artifacts from one project in Jira to another project in Jira), the best practice is to create two separate repository connections in Tasktop for the source repository and the target repository. This will eliminate errors encountered in Tasktop related to relationship fields.

## Synchronizing External Relationships

If you'd like a more lightweight approach, you can configure the scenario below to flow the URL of the related artifact in the source repository to a weblink or string field in the target repository. This is what we refer to as **External ARM** (Artifact Relationship Management).



**External Artifact Relationship Management (ARM)**  
The ability to maintain relationships between artifacts by flowing a URL for the related artifact to a string or weblink field.

Both internal ARM and external ARM are configured the same way with regard to the source collection: A relationship field in the source collection is mapped to a relationship field in the model.

The crucial difference is how the target collection is configured:

- For internal ARM, that relationship field in the model is then mapped to a relationship field in the target collection.
- For external ARM, that relationship field in the model is then mapped to a string field or weblink field in the target collection.

	Source Repository Field	Model Field	Target Repository Field
<b>Internal</b> Artifact Relationship Management (ARM)	Relationship Field	Relationship Field	Relationship Field
<b>External</b> Artifact Relationship Management (ARM)	Relationship Field	Relationship Field	String / Weblink Field

To configure External ARM in Tasktop, follow the instructions below:

**Note:** First, confirm that both repositories support the following in our [Connector Documentation](#):

For the source repository:

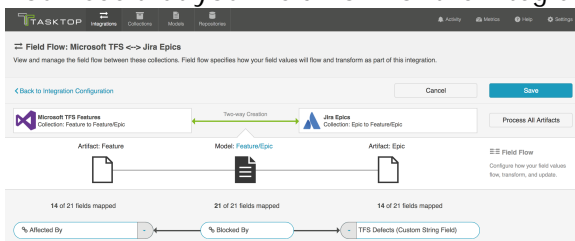
- Relationship field types are supported
- The related artifact type (whose URL you would like to flow) is supported, and provides a unique URL

For the target repository:

- String fields or weblink fields are supported

### Instructions

1. Here, our goal will be similar to the goal in the Internal ARM section: to flow Microsoft TFS Features to Jira Epics. For any TFS Features that have related TFS Defects, instead of creating a related defect in Jira, we'd like to flow the URL for each defect to a custom string field on the Jira Epic.
2. In this scenario, we will only configure 2 collections (Microsoft TFS Features and Jira Epics), and 1 integration (Microsoft TFS Features Jira Epics), in contrast to the internal ARM scenario, which required two integrations. A second integration is not needed here, because we are not **creating** target defects in Jira. Rather, we are flowing the URL of the source defect to a custom field on the Jira Epic.
3. To configure this scenario, create a synchronize integration for your main artifact type.
  - a. In this example, we will flow Microsoft TFS Features to Jira Epics.
4. On the source collection (Microsoft TFS Features), configure a relationship mapping for the relationship type you'd like to flow.
  - a. In this example, we will map "Affected by" relationship field to our 'blocked by' relationship field in the model.
5. On the target collection (Jira Epics), configure a mapping between the string or weblink field that you'd like to receive the URL, and the relationship field in the model that was mapped in the prior step.
  - a. In this example, we will map the Jira custom string field, "TFS Defects" to the "blocked" relationship field in the model.
6. You'll see that your field flow for the integration looks like this:



7. When we run our integration, we will see that Microsoft TFS Features create Epics in Jira, AND that the related defects in Microsoft TFS flow their URLs to the Web Links field on the Jira Epic.

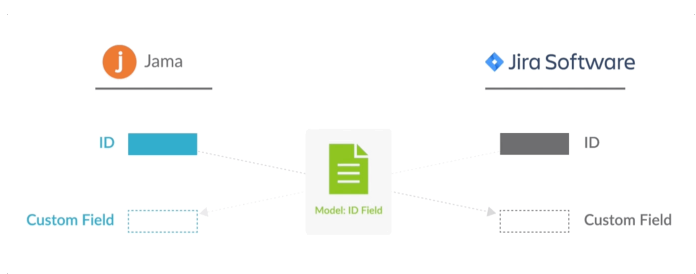
# Synchronizing an Artifact ID or URL Reference

## Synchronizing an Artifact ID or URL Reference

Imagine this scenario: You are flowing defects between two repositories: Jira and Jama. You'd like to have a way to know the ID, or URL, of the source artifact in Jira when viewing its target artifact in Jama (and vice versa). This will provide traceability between the source artifacts and the artifacts that have been created in your target repositories via your integration.

To set this up, you will need to configure two different field mappings in each collection:

- You will need to specify which field to pull the source artifact's ID (or URL) from
- You will need to specify which field to use to store the source artifact's ID (or URL), in your target repository



In the diagram above, you can see that Jira is flowing its ID field to a custom field in Jama, and that Jama is flowing its ID field to a custom field in Jira. In order to set up this integration, you will need to configure your model to accept that ID field. We'll walk through how to do that below.

The instructions below will walk you through how to set up this configuration for the ID field, but the same instructions will also apply for location/URL:

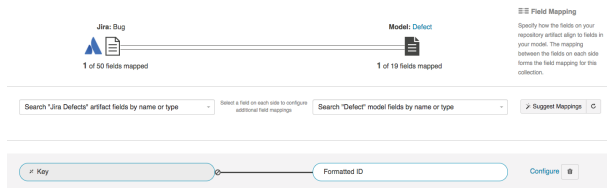
1. Go to the **Model** that you are utilizing in the integration. Ensure that your model includes the Formatted ID field.

We've also shown the 'Location' field below, for reference, as a similar process can be followed to flow the source artifact's URL to a field on the target artifact, for traceability.

Standard Field	Label	Type	Required	
Formatted ID	Formatted ID	String	<input type="checkbox"/>	⊕ ▼
Location	Location	Location	<input type="checkbox"/>	⊕ ▲

2. Go to the **Collections** screen for each of your repositories, and set up mapping to tell the integration where to pull the ID from:

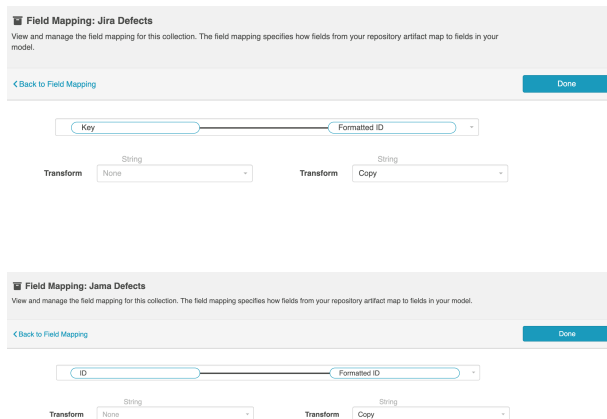
- a. Map the Formatted ID model field to the corresponding field in your repository. This is the field that the collection will take the ID data from. Note that Formatted ID is called 'Key' in Jira, but may be referred to using a different name in a different repository (i.e. 'issue ID')



- b. Click 'Configure' next to your mapping, and confirm that your Transforms are configured as shown below. The transform on the left should be 'None' and the transform on the right should be 'Copy.' This will tell the collection to *send* data from the Key field in your repository to the model, but not vice versa.

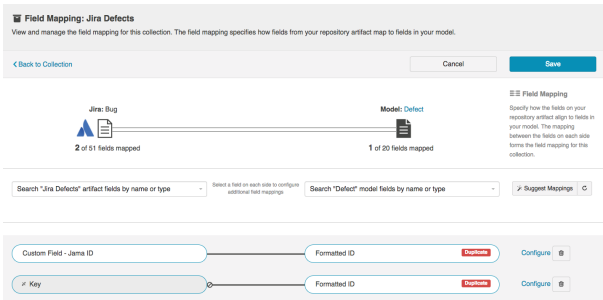


- c. Repeat these steps in your other repository.
- d. Here is how the mappings should look in each repository, for your *source* fields:



3. Now that our model is able to acquire ID data from each source repository, let's tell it where to store that data in the corresponding target repository. To do this, you will set up an additional mapping in each **Collection**:

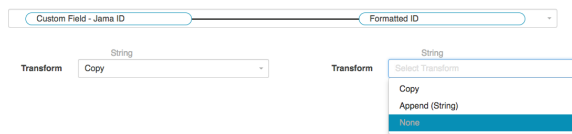
- a. Navigate to one of your **Collections**.
- b. Map the Formatted ID model field to your repository once more, this time to determine where you would like to **store** this data in your target repository. The field mapping page will tell you that this is a 'duplicate,' but that is ok!



In the image above, we have mapped 'formatted ID' to a custom field in Jira called 'Custom Field - Jama ID'. This is the field that the Jama Formatted ID data will flow to in Jira.

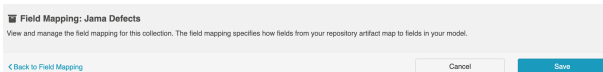
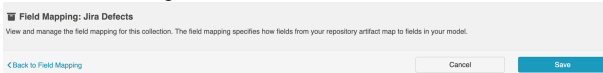
💡 Note: Do not click 'Save' yet. If you do, you will get an error. Continue to the next step below.

- c. Click 'Configure' on the new mapping, and configure as shown below. This will tell the collection to take data from the model and send it to the 'Description' field, but not vice versa.



💡 Note: The transform on the left may be 'Copy,' 'Formatted String to Rich Text,' or some other transform depending on the field types of the repository field and model field. However, the important thing is that the transform on the right (on the model side) be set to 'None.' This ensures that data will only flow *into* the repository field, rather than *out* of it.

- d. Save your mapping and collection.
- e. Repeat these steps on your other collection.
- f. Here is how your transforms should look in each collection, for your *target* fields:



4. When you run the integration, the ID of the source artifact will now flow to a field on the target artifact (and vice versa), as specified in your field mapping:

The image shows two screenshots illustrating the integration of Jira and Rally artifacts. The top screenshot shows a Jira issue titled "Welcome Page is missing the top banner" with a custom field "Jama Flight-BUG-1 ID" containing the value "FRA-1". The bottom screenshot shows the Rally interface for a defect with the same title, where the "ID" field is "Flight-BUG-1" and the "Custom Field - Jira ID" field is "FRA-1".

**Jira Issue Details:**

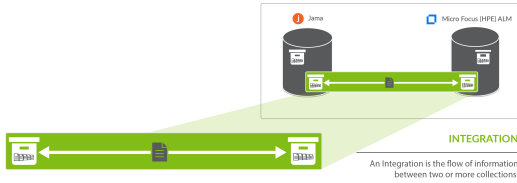
- Issue ID: FRA-1
- Title: Welcome Page is missing the top banner
- Type: Bug
- Priority: Medium
- Status: TO DO (View workflow)
- Resolution: Unresolved
- Labels: None
- Custom Field - Jama Flight-BUG-1 ID: FRA-1
- Description: Welcome Page is missing the top banner

**Rally Defect Details:**

- Defect ID: Flight-BUG-1
- Global ID: GID-31303
- Name: Welcome page is missing the top banner
- Description: Welcome page is missing the top banner
- Release: Unassigned
- Priority: Unassigned
- Found By: Unassigned
- Assigned: Unassigned
- extra text field: Unassigned
- Proxy Attribute: Unassigned
- ProxyStorage: Unassigned
- Status: New
- fixversion: Unassigned
- ALM Proj: Unassigned
- Alternate ID: Unassigned
- Alternate URL: Unassigned
- Test Model Update: Unassigned
- Custom Field - Jira ID: FRA-1

# Container + Work Item Synchronization

## What is an Integration?



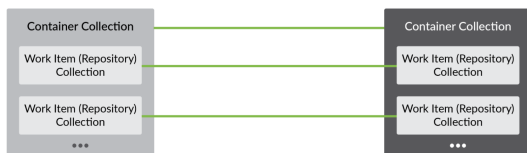
An *integration* is quite simply **the flow of information between two or more collections**. When you configure your integration, you can customize the field flow, artifact routing, artifact filtering, as well as enable or disable comment flow or attachment flow.

## What is a Container + Work Item Synchronization?

The Container + Work Item Synchronization template enables you to flow your folder structure from one repository to the other, along with any corresponding work items (such as defects, requirements, etc) that are contained within that structure. The term **folder** is used loosely, and can refer to many container types, such as folders, modules, or packages.

## Template Affordances

The Container + Work Item Synchronization template allows you to flow containers and their contained work items between two repositories. The integration will consist of two container collections and two (or more) work item collections from the same repositories.

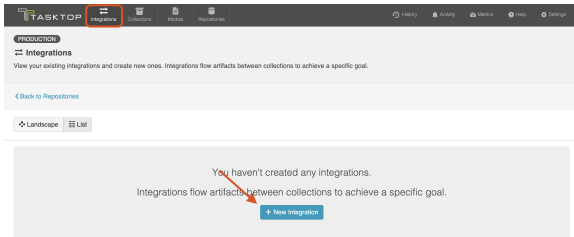


## Configuring a Container + Work Item Synchronization Integration

Once you have your base repositories and collections set up, you can configure integrations to synchronize the artifacts in those collections.

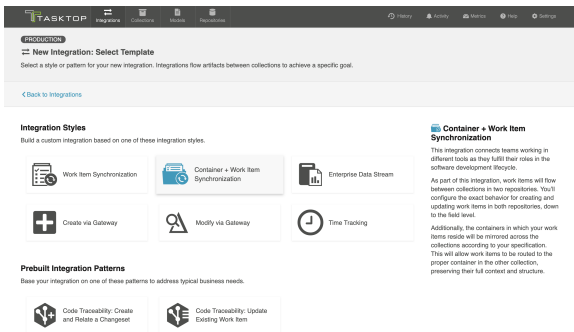
In this scenario, we'll show you how to configure an integration that flows containers (folders) along with the work items (requirements) contained within them, from a source repository to a target repository.

To configure your integration, select **Integrations** at the top of the screen, then click **+ New Integration**.

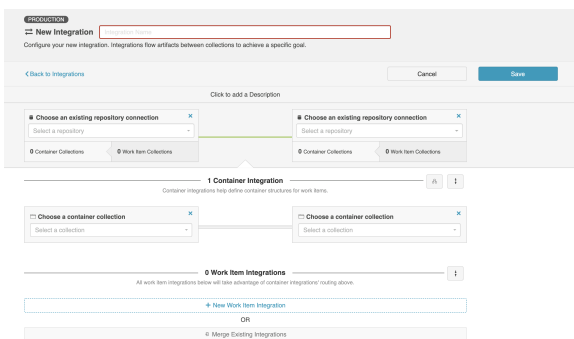


Select the **Container + Work Item Synchronization** integration template.

💡 Depending on the **edition** of Tasktop you are utilizing, you may not have all options shown here.



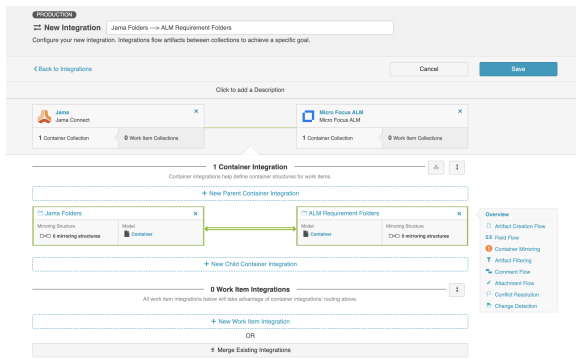
This will bring you to the **New Integration** screen:



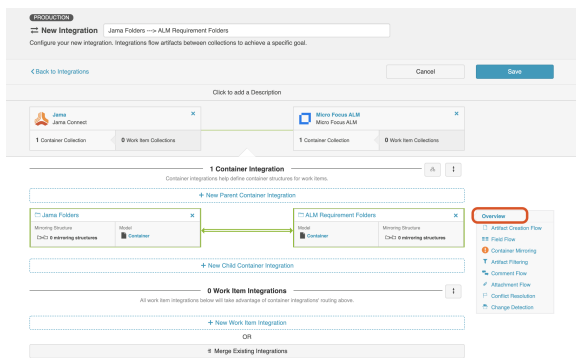
Name your integration and select your repositories and container collections, and then click **Save**.

💡 **Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your integration.





You can click the **Overview** link on the right side of the Integration Configuration screen to get to the main display page.



From this page, you can configure many different components of your work item synchronization.

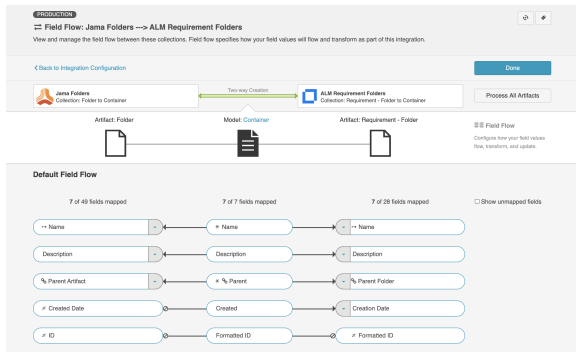
Configuring your Container Integration is very similar to configuring a [Work Item Synchronization](#). Please refer to that page for details, while taking note of the key differences outlined below.

## Artifact Creation Flow

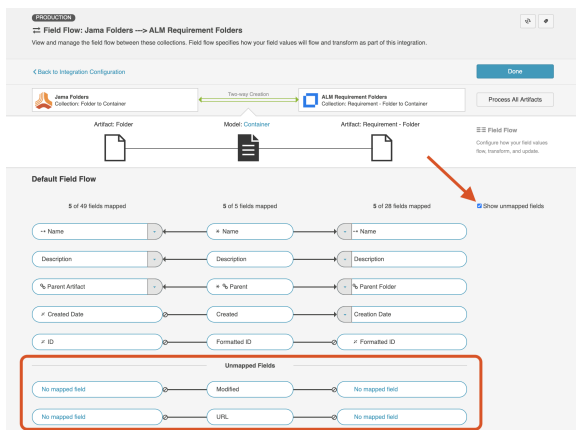
This process is the same as it is for a Work Item Synchronization. Refer to the [Artifact Creation Flow](#) page for details.

## Field Flow

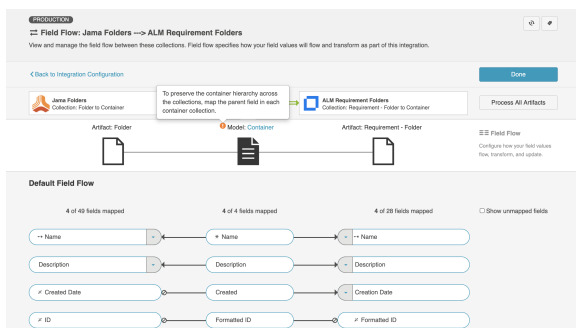
Similar to a Work Item Synchronization, you can click **Field Flow** to configure how fields will flow in your Container Integration. Typically, container integrations will flow significantly fewer fields than a work item integration.



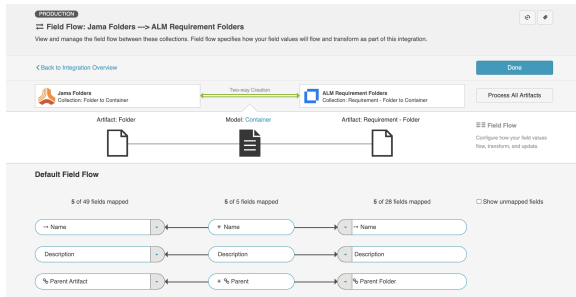
From the Field Flow screen, you can see the names of the mapped repository fields for each collection on the far left and right, with the model fields displayed in the middle. By default, model fields without mapped repository fields are hidden. You can see all model fields by toggling the **Show unmapped fields** checkbox. Constant values will be identified by a gray box and the constant value icon.



You may also notice a warning reminding you to map the parent field in each container collection. Doing so will ensure that nested containers flow to your target collection along with the appropriate hierarchical structure.




Once you map the Parent field in each collection appropriately, you'll see that the warning disappears.



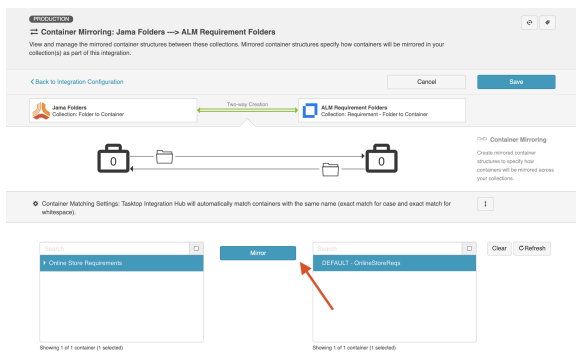
## Container Mirroring

Container Mirroring is similar to the concept of Artifact Routing (within a Work Item Synchronization), but it has some key differences.

On the Container Mirroring screen, you'll see the hierarchical organizational structure contained within each collection. Select the desired top level container on each side. Once joined, Tasktop will know to mirror the container structure underneath in the target collection.

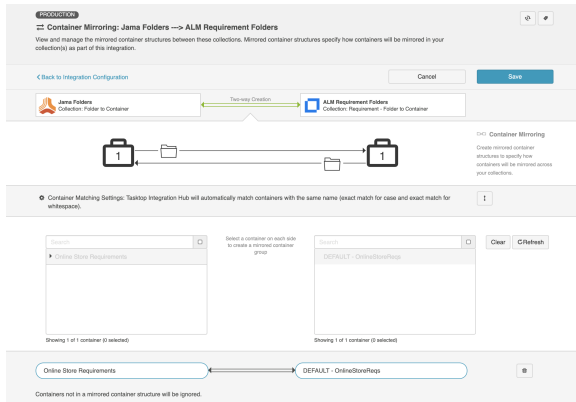
 Note that the container structure underneath the top level container will not display in Tasktop unless those container levels can also serve as 'top level containers' for the purposes of mirroring.

Unlike Artifact Routing, Container Mirroring pairs must be one-to-one.

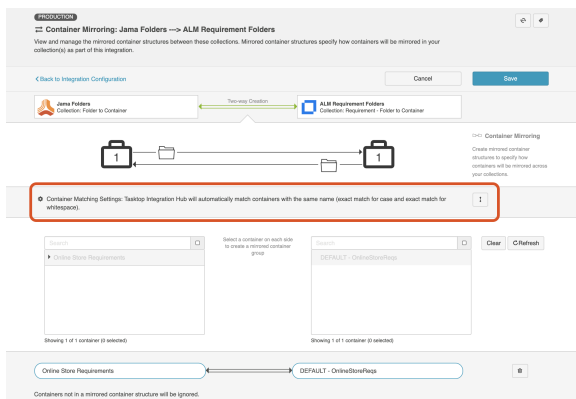


In the example above, any folders contained within the Online Store Requirements project in Jama will create corresponding folders in the Online Store Requirements project in Micro Focus ALM, and vice versa.

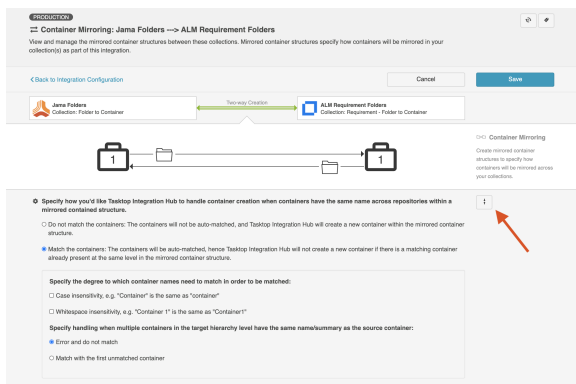
Once you've completed mapping your mirrored pairs, you'll see them in the grey sash below:



You'll also notice a **Container Matching Settings** sash.



Click the **expand** button to configure your Container Matching settings.



If you choose to **match the containers**, Tasktop will proactively find any existing containers that have the same name (summary) across collections (so long as they are in the same level of the mirrored container structure) and match them. When Tasktop **matches** two containers:

- No new container will be created in the target repository, as a **matched** container already exists.
- Any work items contained within the matched containers will route to one another, unless the corresponding work item integration's artifact routing overrides that route.
- Any sub-containers beneath the matched containers will mirror one another.

- An event of type, **associated artifacts**, will be displayed on the Activity screen indicating that the two containers were matched.

You will also be able to specify whether you'd like your matching strategy to be case sensitive or whitespace sensitive, and to specify how Tasktop should handle situations where there are multiple containers in the target hierarchy level that have the same name/summary as the source container.

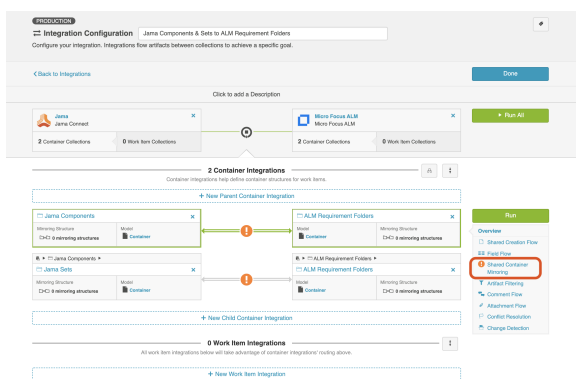
When configuring a new integration, the container matching settings will default to **match the containers with error and do not match** selected.

## Shared Container Mirroring

If your container integration has mismatched hierarchical structures (e.g., Jama has components, sets, and folders and ALM only has groups of nested folders), you can synchronize several types of containers to one type of container using the shared container mirroring configuration option.

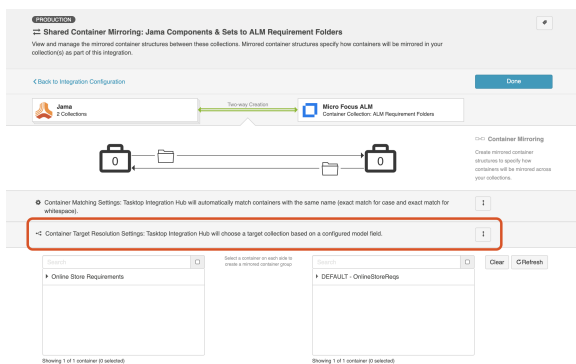
To configure shared container mirroring, set up a container integration where the same collection type is reused in multiple levels of the integration.

You'll then see the **Shared Container Mirroring** link on the **Integration Configuration** screen.



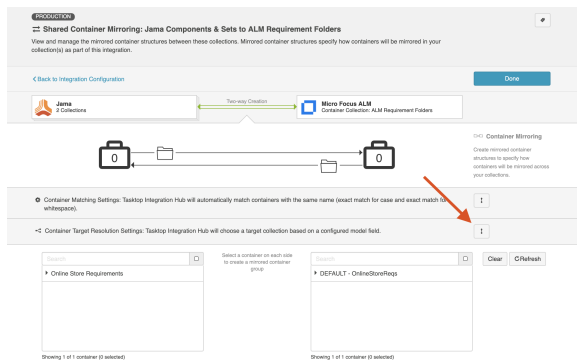
The **Shared Container Mirroring** screen is similar to the Container Mirroring screen, with the exception of **Container Target Resolution Settings**. This setting will appear if artifact creation flow is configured to flow away from containers sharing a type.

**Note:** If this setting appears on the Shared Container Mirroring, it **must** be configured for artifacts to flow in your integration.



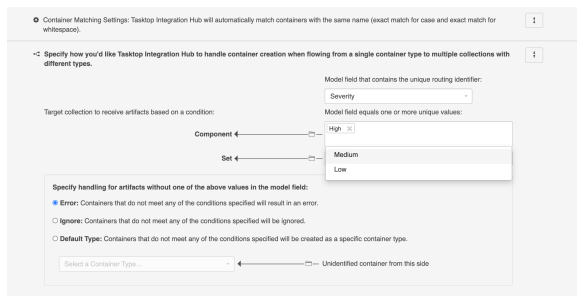
Click the **expand** button to configure your **Container Target Resolution** settings.

**Note:** When configuring a new integration, the container target resolution settings will default to **error** unless otherwise specified.

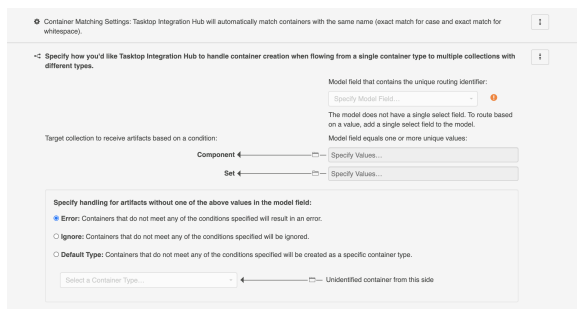


To configure container target resolution settings, select a model field and choose which collection to flow into based on values for the selected field.

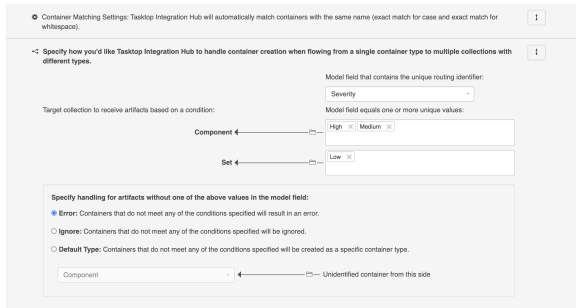
**Note:** The model field must be a single select field and must have options configured in the model. The selected model field must also be mapped bidirectionally.



If the model doesn't contain a single select field, you'll notice a warning next to the model field dropdown.



Once a single select value is added to the model, you'll see that the warning disappears.



You'll also see that you can specify handling for artifacts without one of the selected values in the model field. You can select from the following options:

- **Error:** Containers that do not meet any of the specified conditions will result in an error.
- **Ignore:** Containers that do not meet any of the conditions specified will be ignored.
- **Default Type:** Containers that do not meet any of the conditions specified will be created as a specific container type.

**Note:** When the target resolution field is configured, it's best to set the field once on a new container item so that it flows to the correct type the first time. Toggling the field to switch the type later on may lead to errors.

## Artifact Filtering

This process is the same as it is for a Work Item Synchronization. Refer to the [Artifact Filtering](#) page for details.

## Comment Flow

This process is the same as it is for a Work Item Synchronization. Refer to the [Comment Flow](#) page for details.

## Attachment Flow

This process is the same as it is for a Work Item Synchronization. Refer to the [Attachment Flow](#) page for details.

## Conflict Resolution

This process is the same as it is for a Work Item Synchronization. Refer to the [Conflict Resolution](#) page for details.

## Change Detection

This process is the same as it is for a Work Item Synchronization. Refer to the [Change Detection](#) page for details.

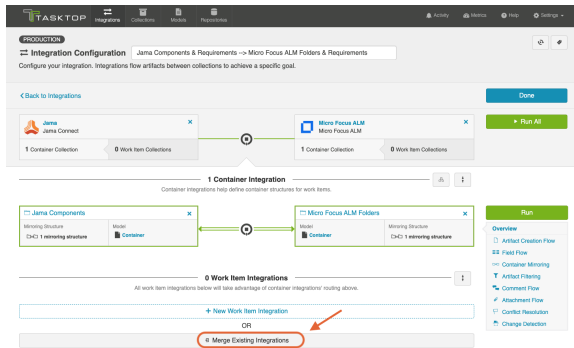
# Configuring your Work Item Integration(s)

To add your Work Item Integration(s), you have two options:

1. Creating a new Work Item Integration from this screen
2. Importing an existing Work Item Integration

## Creating a New Work Item Integration

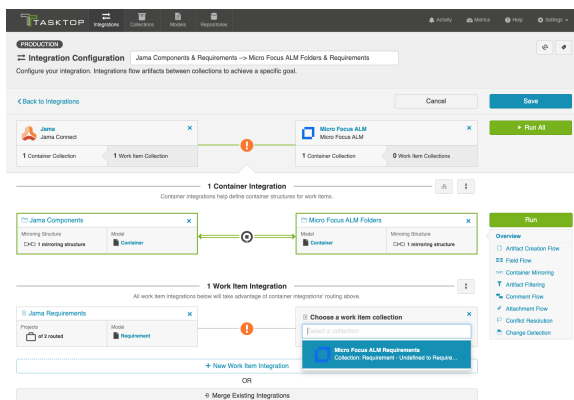
To create a new Work Item Integration, click **+ New Work Item Integration**.



You will be prompted to select the existing work item collections you'd like to add to the integration.

To add a work item collection to the integration, it must:

- Be from the same repositories as the container integration above
- Include work item types that can take advantage of container mirroring (for example, in the scenario below, we will not be able to add a Micro Focus Defects collection, since only requirements can be routed to Micro Focus requirements folders)



Once added, click **Save**.



In general, you will configure this in the exact same way you configure a normal [Work Item Synchronization](#), with just a couple of key differences with regard to Artifact Routing outlined in the Artifact Routing Section, below. Please refer to the [Work Item Synchronization](#) page for details on all other aspects of configuration.

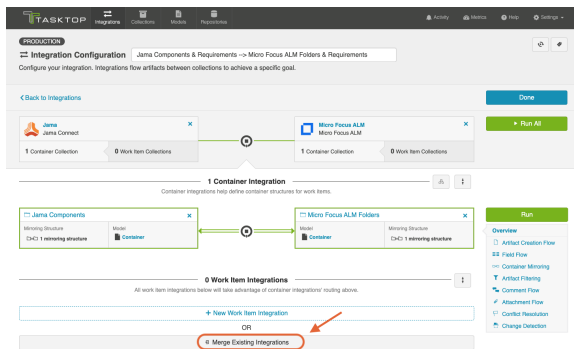
## Merging an Existing Work Item Integration

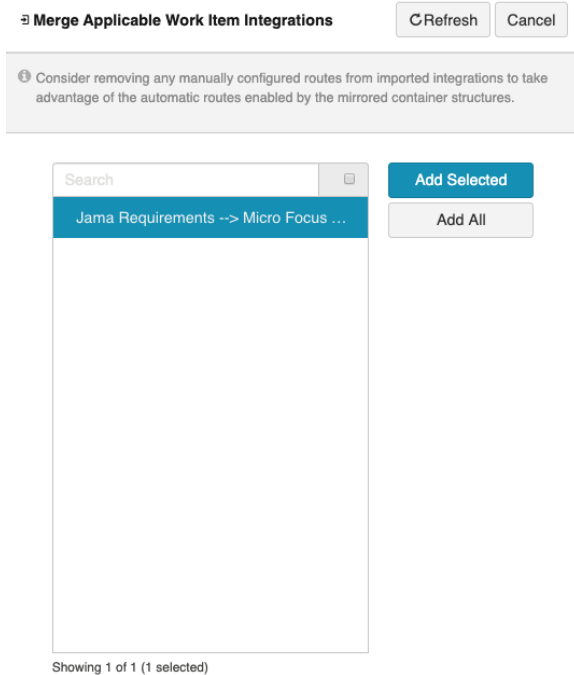
If you've already configured a Work Item Synchronization that you'd like to run as part of this integration, you can add it by clicking **Merge Existing Integrations**.

**⚠️ Note** that once you merge your integration, it will cease to exist as an independent integration. You will only be able to access and configure it from this Work Item + Container Mirroring integration.

To merge an existing integration, it must:

- Be from the same repositories as the container integration above
  - **💡 Note** that the order matters (i.e., if the work item integration reverses which repository is on the left vs. right side, an error will occur). For this reason, it is very important to ensure that integrations are created consistently with regard to which repository is on each side.
- Include work item types that can take advantage of container mirroring (e.g., in the scenario below, we will not be able to add a Micro Focus Defects integration, since only requirements can be routed to Micro Focus requirements folders).





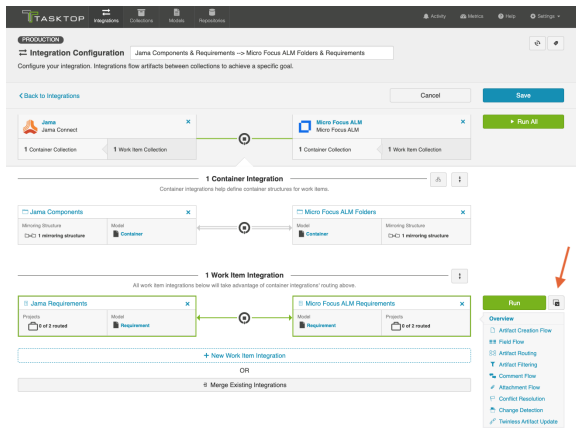
When merging an existing integration, consider removing any manually configured routes from that imported integration to allow it to take advantage of the automatic routes enabled by the mirrored container structures.

After clicking **Add Selected**, you'll see that integration added to the Integration Configuration screen.

⚠ If you'd like to detach the integration, follow the steps outlined in the **Detaching a Work Item Integration** section below. Do not click the **x**'s in the upper right corner of each collection, as this will remove those collections (along with any associated configuration, such as Artifact Routing) from the integration permanently. Since the merged Work Item Synchronization only exists as part of the Container + Work Item Synchronization, any changes you make to that integration here will be permanent.

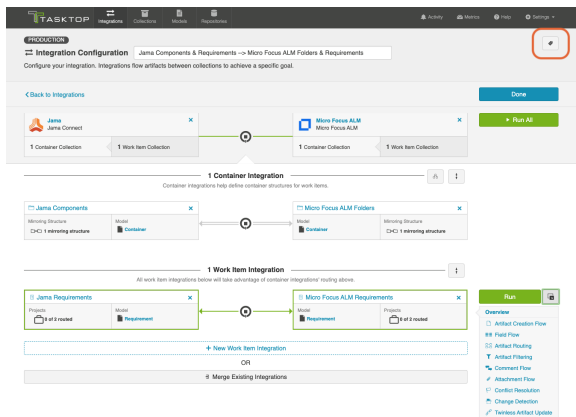
## Activating the Configuration Pane

To activate the configuration pane for the integration you'd like to modify, highlight the integration by clicking its arrow. This will enable the configuration links for that particular integration.



## Viewing Associated Configuration Elements

To view associated configuration elements (such as collections or models that utilize the integration you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



### Associated Elements for Integration "Jama Components & Requirements -> Micro Focus ALM Folders & Requirements"

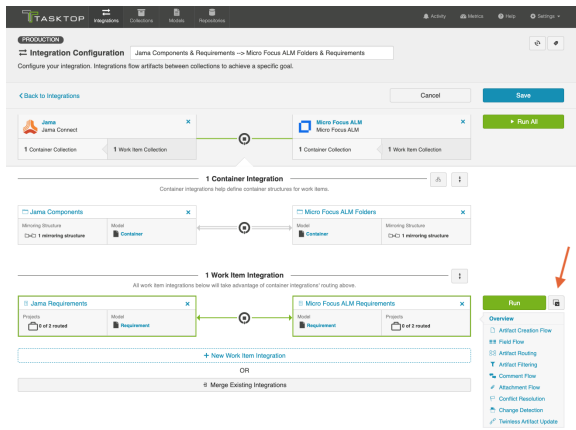
- **2 Models used by this Integration**
  - [Container](#)
  - [Requirement](#)
- **4 Repository Collections used by this Integration**
  - [Jama Components](#)
  - [Jama Requirements](#)
  - [Micro Focus ALM Folders](#)
  - [Micro Focus ALM Requirements](#)
- **2 Repository Connections used by this Integration**
  - [Jama](#)
  - [Micro Focus ALM](#)

Close

## Detaching a Work Item Integration

If you'd like to detach a Work Item Integration (so that it exists as an independent integration, accessible from the Integrations List page, rather than as part of this Work Item + Container Mirroring Integration), make sure the configuration pane for that integration is enabled (see steps above).

Next, click the **Detach** button.



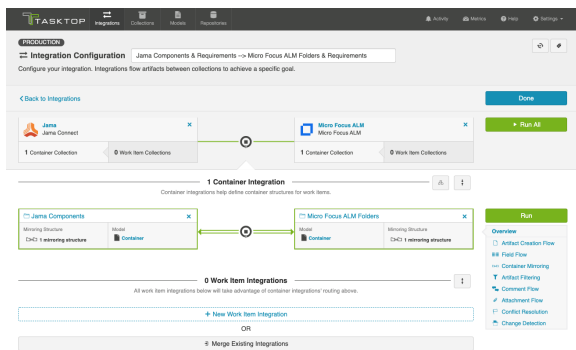
You will be prompted to name your integration:

#### △ Detach Integration

This will remove the work item synchronization from this work item synchronization + container mirroring integration. From this point forward, the work items in the detached synchronization will not be able to take advantage of the automatic routes enabled by the mirrored container structures in this integration.

To detach this work item synchronization, please specify a name for it.

You'll notice that the integration is no longer included as part of this Container + Work Item Synchronization:



You'll also notice that you can now access that integration from the Integration List view:

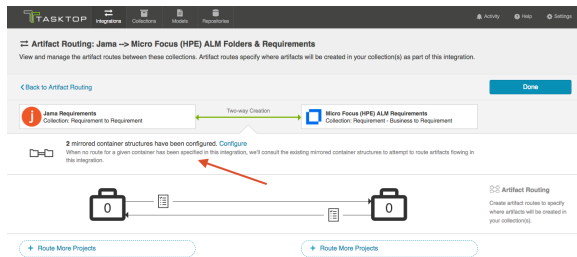


## Configuring Your Work Item Integration

In general, configuration for the Work Item Integration contained within your Container + Work Item Synchronization will be very similar to configuration for a typical [Work Item Synchronization](#), with the exception of a few key differences, outlined below. Please refer to the [Work Item Synchronization](#) page for details on all other aspects of configuration.

## Artifact Routing

On the Artifact Routing screen for your Work Item Integration, you will see a reference to the existing Container Mirroring configuration that was set up as part of the Container Integration.



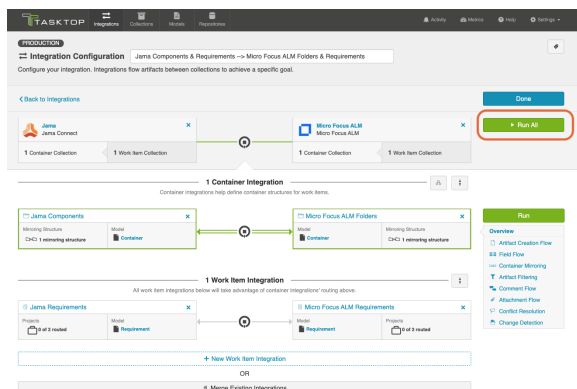
Where applicable, your work items will flow in accordance with the Container Mirroring that has been configured. In addition to the routing that is inherited based on Container Mirroring, Artifact Routing can be configured on this page to determine where work items will flow with regard to containers not included in the Container Mirroring structure. If you configure Artifact Routing that contradicts the Container Mirroring configuration, the Artifact Routing configuration will take precedence when determining how work items will flow.

**Note:** If you would like your artifact routing to match your container mirroring, but to only flow artifacts from a subset of those containers, that use case cannot be accommodated from the Artifact Routing screen here. To satisfy this use case, you will need to detach your work item integration from the Container + Work Item Synchronization. Once detached, you can configure Artifact Routing for the independent work item integration to successfully limit the containers utilized.

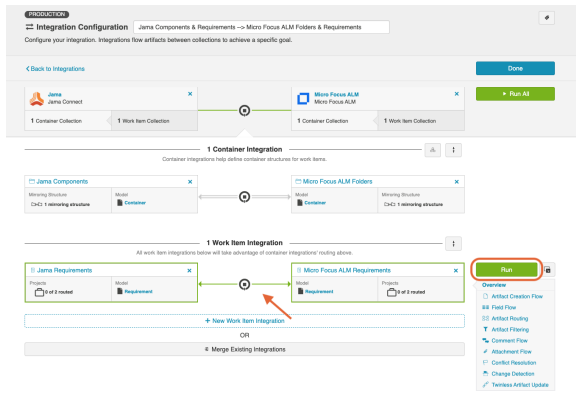
## Running your Integration

**Warning:** Please be aware that integrations will trigger changes in your end repositories and that misconfiguration can cause items to be duplicated or modified in unexpected ways. Additionally, there is no **undo** in Tasktop or the repositories. If you have any questions, please [contact support](#).

Since your Container + Work Item Synchronization technically consists of several independent, but interconnected integrations, you can select **Run All** to run all integrations at once, or choose to run integrations independently.



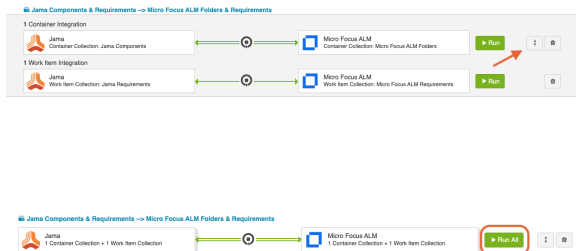
If for any reason you'd like to run an integration individually, activate that integration's configuration pane by clicking on its arrows, and then click **Run**.



You can also view and run your integration(s) from the Integration List screen. On this screen, your integration will default to the expanded view, where you can run each integration individually:



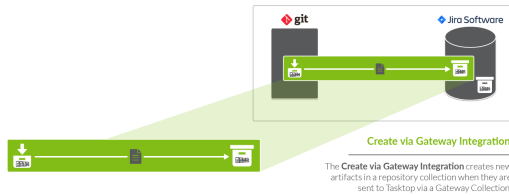
If you'd like to **Run All**, you can collapse the view and then click **Run All**:



# Create via Gateway

The Create via Gateway Integration Template is only available in Editions that contain the Gateway add-on. See [Tasktop Editions table](#) to determine if your edition contains this functionality.

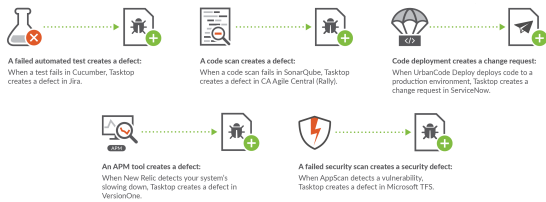
## What is a Create via Gateway Integration?



An *integration* is quite simply **the flow of information between two or more collections**. A *Create via Gateway Integration*, specifically, creates new artifacts in a work item collection or a container collection that connects to a repository, such as Jira, when they are sent to Tasktop via a Gateway Collection. The Gateway Collection uses an inbound webhook to access event-based information in an external DevOps tool, such as Git or Jenkins.

These types of events are “fire and forget” - they create something new in your repository, but they don’t expect anything back. As such, they don’t mandate a full-blown two way synchronization; a lighter one-way integration can do the trick.


Here are some examples of what you can do with the Create via Gateway integration template:



When you configure a Create via Gateway Integration, you can customize the field flow, artifact routing, and artifact filtering of your integration.

## Video Tutorial

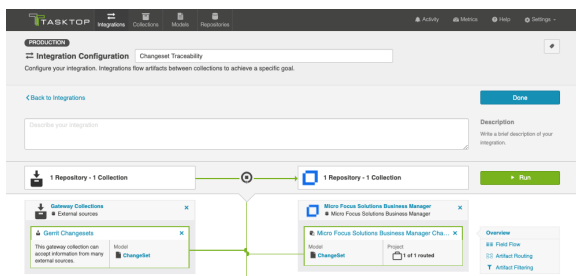
Check out the video below to learn how to configure the Create via Gateway Integration Template.

 This video assumes that you have already configured your repositories, models, and collections as outlined in the [Quick Start Guide](#).

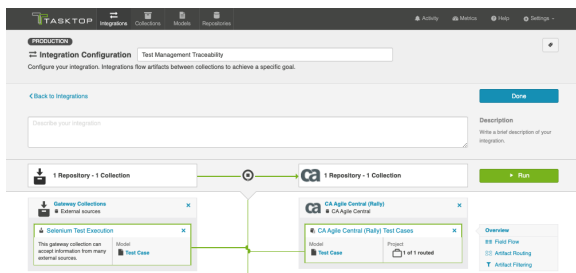
## Use Case and Business Value

The Create via Gateway integration creates traceability between artifacts across the software development lifecycle. New artifacts will be created in a work item (repository) collection or container (repository) collection when artifacts are sent to Tasktop via a gateway collection. Optionally, these newly-created artifacts can be related to already-existing artifacts in the same repository.

For example, if your development team uses Gerrit for source code management and Solutions Business Manager (formerly Serena Business Manager) for Agile story management, you can set up an integration that would trigger the creation of changesets in SBM whenever changesets were created in Gerrit. And if the changesets in Gerrit identify the stories in SBM to which they pertain, Tasktop would find the already-existing story in SBM and create a relationship between that story and the newly created changeset in SBM.



Additionally, if your QA team uses a tool like Selenium for test execution, and a tool like CA Agile Central (Rally) for test management, you can set up an integration that would trigger the creation of test results in CA Agile Central (Rally) when test results are created in Selenium. And if the test results from Selenium identify the tests in CA Agile Central (Rally) which they cover, Tasktop would find the already-existing test and create a relationship between the two artifacts.



## Template Affordances

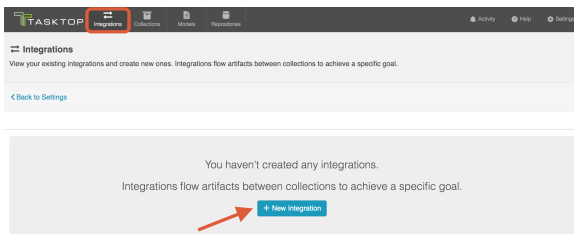


The Create via Gateway Integration Template allows you to flow artifacts from a single gateway collection into a single work item or container collection that connects to a repository. When a new artifact is sent to Tasktop via our REST API, an artifact will be created in the target work item or container collection.



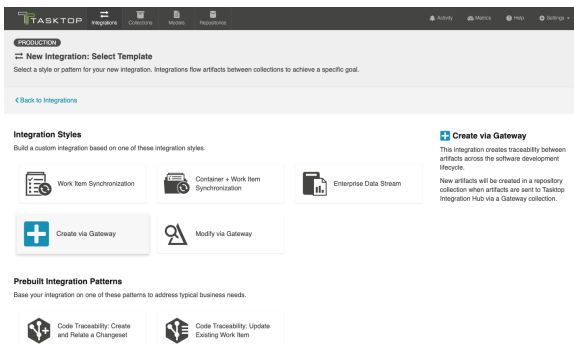
## Configuring a Create via Gateway Integration

To configure your integration, select **Integrations** at the top of the screen, then click **New Integration**.

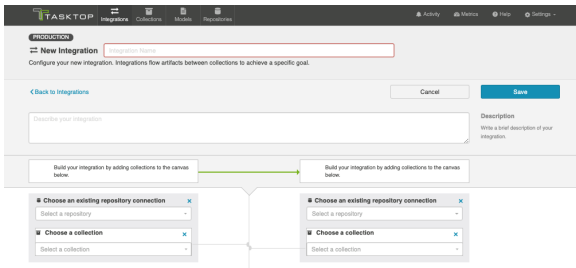


Select the **Create via Gateway** template.

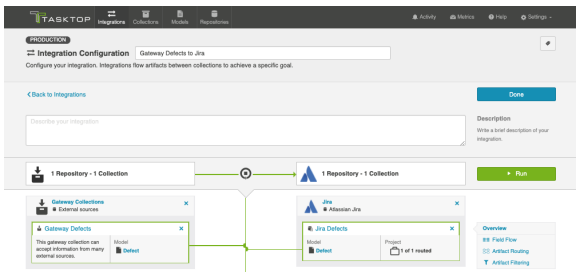
💡 Depending on the [edition](#) of Tasktop you are utilizing, you may not see all options shown below.



This will bring you to the New Integration screen.

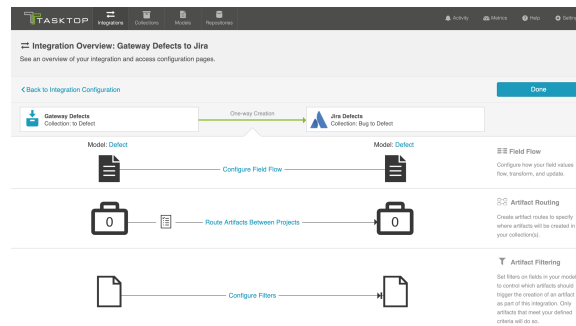
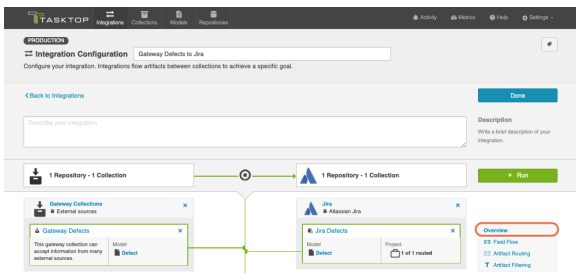


Name your integration and select your repositories and collections.



You'll notice a configuration warning next to the Artifact Routing link if you haven't configured your routing yet. Routing is essential, since it tells your integration *where* (in which project, for example) to create each artifact's twin. You can learn more about Artifact Routing [below](#).

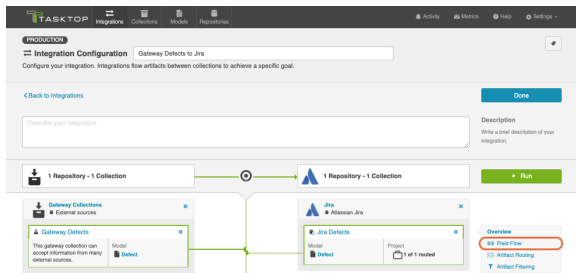
You can click the **Overview** link on the right side of the Integration screen to get to the Overview screen.



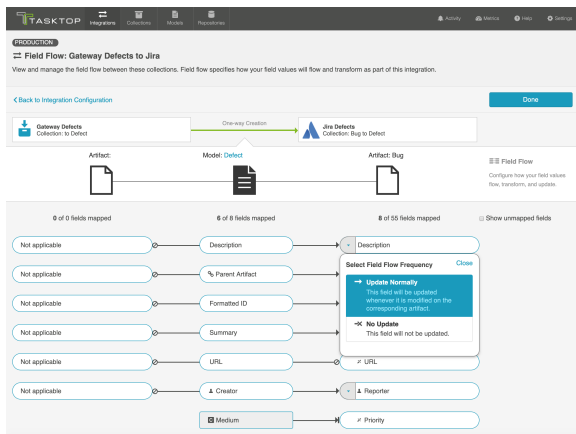
## Field Flow

The field flow configured for a given integration specifies which fields should flow in that integration. For Create via Gateway integrations, you can choose to flow a given field (Update Normally) or to not flow a given field (No Update).

To get to the Field Flow screen, click **Field Flow** on the right pane of the Integration Configuration screen.



You will be directed to the Field Flow screen.








You can choose to flow a field ('update normally') or not flow it ('no update'). You'll notice that field flow goes in one direction only — from the gateway collection *into* the repository or database collection.

You can see the names of the mapped artifact fields for each collection on the far left and far right, with the model fields displayed in the middle. By default, model fields without mapped repository fields are hidden. You can see all model fields by checking the **Show unmapped fields** checkbox. Constant values will be identified by a grey box and the constant value icon.

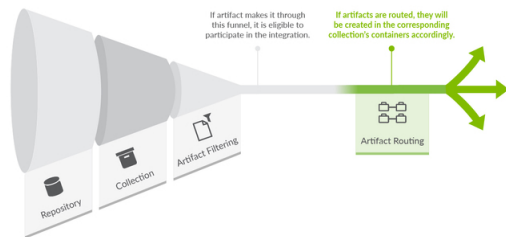
## Field Flow Icons

On the Field Flow screen, you will see a number of icons, which will help you understand any special properties or requirements for each field. If you hover your mouse over an icon, you will see a pop-up explaining what the icon means. You can also review their meanings in the legend below:

Icon	Meaning
------	---------

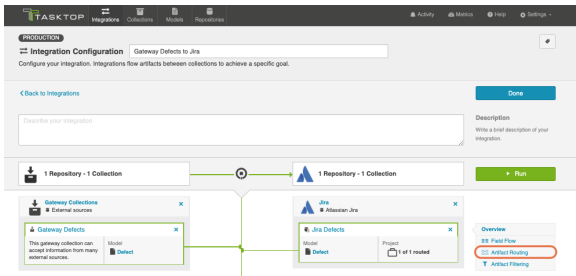
<p><b>C</b></p>	<p>A constant value will be sent.</p> <p>Note that:</p> <ul style="list-style-type: none"> <li>• If the icon is on the side of the collection, this means that a constant value will be sent to your model. This means that any time this collection is integrated with another collection, the <i>other</i> collection will receive this constant value for the field in question.</li> <li>• If the icon is on the side of the model, this means a constant value will be sent to your collection. This means that any time this collection is integrated with another collection, that <i>this</i> collection will receive this constant value for the field in question.</li> </ul>
	<p>Collection field is read-only and cannot receive data</p>
<p>←*</p> <p>*→</p>	<p>To create artifacts in your collection, this field must be mapped to your model.</p>
	<p>This is a required field in your model; it must be mapped to your collection.</p>
	<p>This field will not be updated as part of your integration, due to how you have configured it. This field flow configuration can be changed if you'd like.</p>
	<p>This field will not be updated as part of your integration because the mapping would be invalid. You do not have the option of changing this.</p>
	<p>This field will update normally as part of your synchronize integration; this means it will be updated whenever it is modified on the corresponding artifact.</p>

## Artifact Routing



Artifact Routing is needed when artifacts are being created as part of an integration. In addition to knowing the repository in which artifacts should be created, Tasktop also needs to know which container (i.e., project, module, folder, etc) a given artifact should be created in. Specifying the artifact routing does this.

To configure Artifact Routing, select **Artifact Routing** on the right pane of the Integration Configuration screen.

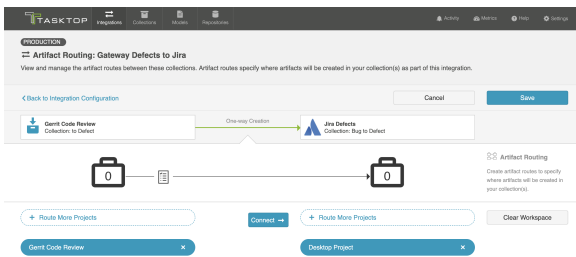


## Static Artifact Routing

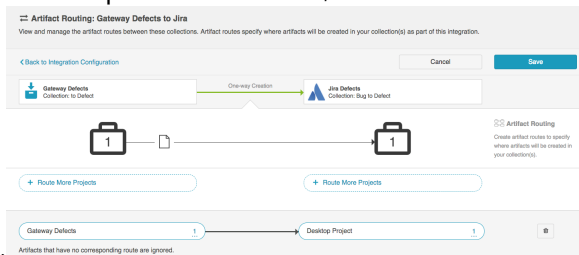
In some cases, all artifacts in a gateway collection are routed to just one project in the target collection. In these instances, you can configure what is known as 'static artifact routing' (also known as 'explicit artifact routing').

To configure a static artifact route, use the **Route More Projects** buttons to add projects from your collections to your workspace and connect them using the **Connect** button.

**Note:** Static artifact routes can have one or more source projects, but only a single target project.



In the example shown below, artifacts from Gateway Defects will be created in the Desktop Project in



Jira. Artifacts that have no corresponding route are ignored.

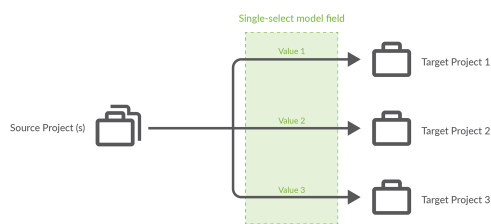
## Conditional Artifact Routing

Check out the video below to learn more about Conditional Artifact Routing:

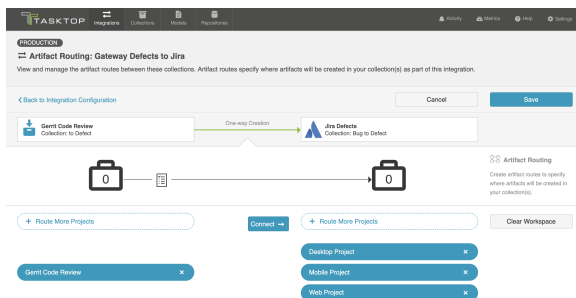
In other cases, you may wish to route your gateway artifacts to multiple projects in the target collection. In this scenario, a field value on the artifact is used to determine which project in the target collection the artifact should route to.

In these instances, you will configure what is known as **conditional artifact routing** to determine which project each artifact is created in within your target repository. Conditional artifact routing (also known as 'dynamic artifact routing') can be used to inspect a single-select field of an artifact and, depending on its value, to route that artifact to the appropriate project in the target collection.

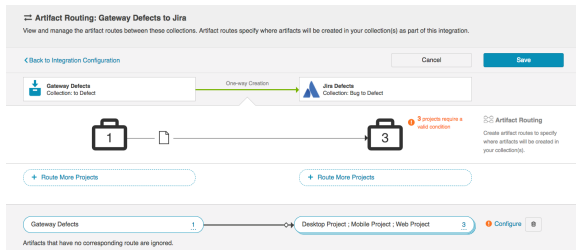
Conditional artifact routes can have one or more source projects, and always have multiple target projects.



To create a conditional artifact route, use the **Route More Projects** buttons to add projects from your collections to your working space and connect them using the **Connect** button.



Notice that after you've created your conditional artifact routing group, you'll be prompted to set the conditions that will define that route.



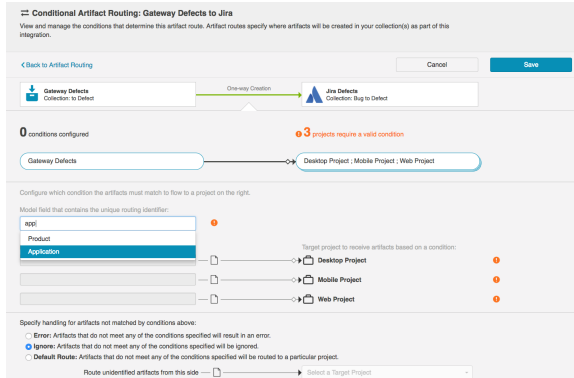
Click **Save** and then click **Configure**.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.

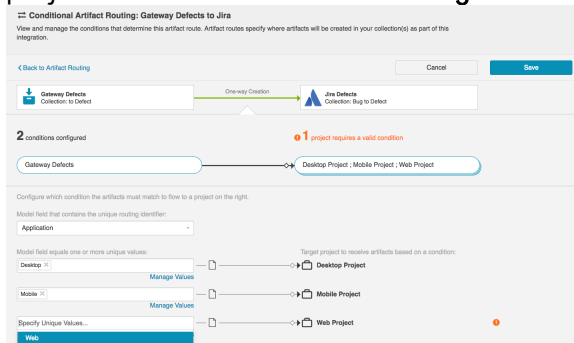
You'll be brought to the Conditional Artifact Routing screen. Here you'll start by selecting the model field on the artifact that you would like to use to determine your artifact route.

**Note:** Conditional Artifact Routes can only be configured based on **single-select fields** in your model.

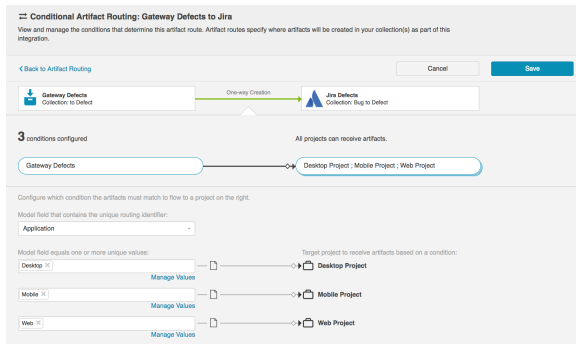
In the example below, the field **Application** contains the unique values that should determine the project an artifact will be created in in Jira.



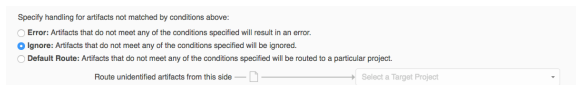
After you select the model field, you can identify one or more value to correspond to each target project. You can also use the **Manage Values** link to select from a list of values.



Once you've done this, you'll see your full conditional artifact routing group.

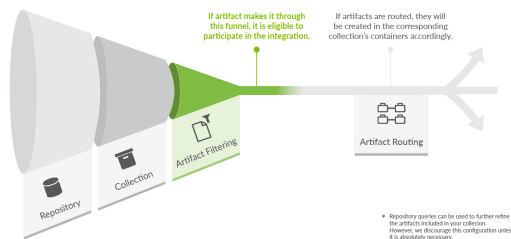


You can also specify how you'd like to handle artifacts that do not meet any of the conditions specified by selecting one of the options provided at the bottom of the screen:



## Artifact Filtering

When configuring your integration, you have several options available to refine which artifacts are eligible to flow. The final mechanism available is *artifact filtering*, which is configured at the Integration level.



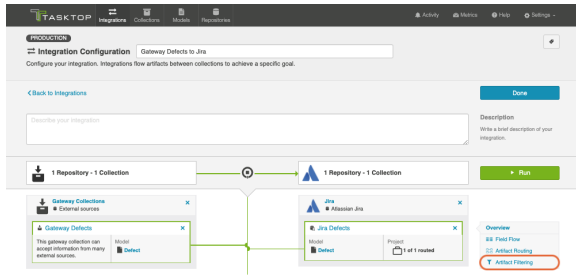
**Artifact Filtering** enables you to set filters in order to limit which artifacts are eligible to flow in an integration.

To use a field for artifact filtering, it must:

- Be a part of your model, and be mapped to the collection you are filtering from
- Be one of the following field types:
  - Single Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **on ly** fields that are already a part of the model will be considered for artifact filtering
  - Multi-Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **on ly** fields that are already a part of the model will be considered for artifact filtering
  - Date
  - Date/Time
  - Duration
  - String

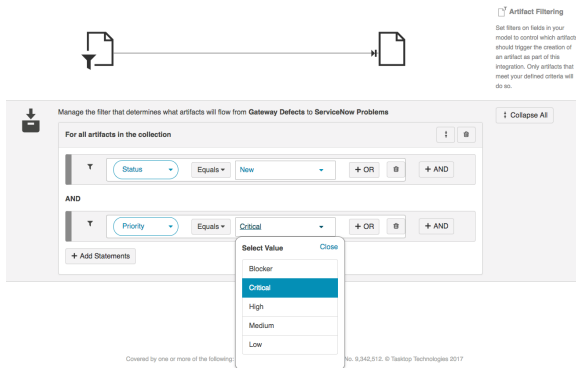


To configure Artifact Filtering, select **Create filters (optional)** from the Integration Configuration Overview screen, or select **Artifact Filtering** from the right pane of the Integration Configuration screen.



This will lead you to the Artifact Filtering Configuration screen, where you can configure one or more criteria for artifact filtering.

💡 You can click the **Collapse All** button to view an easier-to-read summary of your artifact filtering statements.



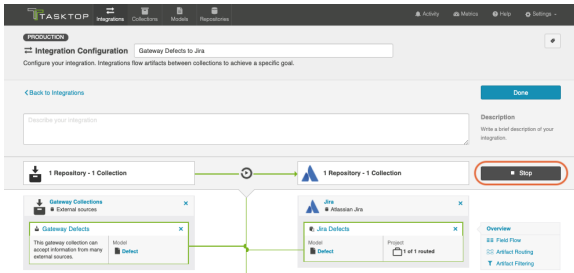
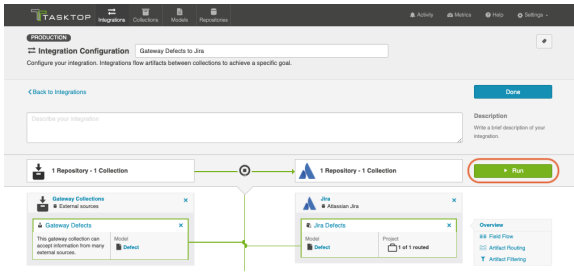
## Running your Integration

⚠️ Please be aware that integrations will trigger changes in your end repositories and that misconfiguration can cause items to be duplicated or modified in unexpected ways. Additionally, there is no 'undo' in Tasktop or the repositories. If you have any questions, please [contact support](#).

There are two ways to start or stop your integration:

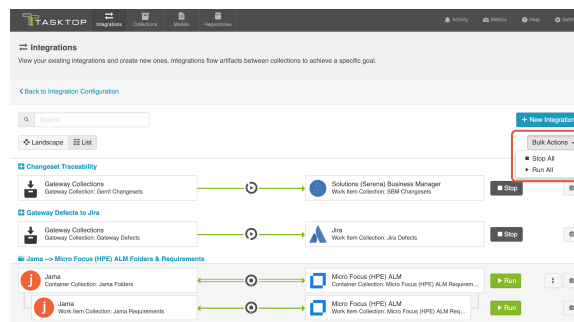
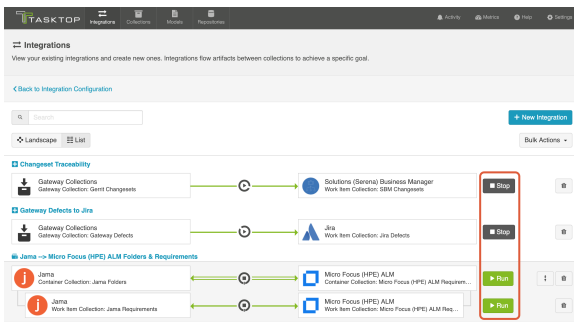
### From the Integration Configuration Screen

Simply click the **Run** button to run the integration, and the **Stop** button to stop the integration.



## From the Integrations List Page

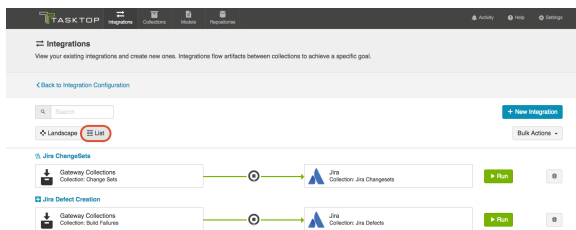
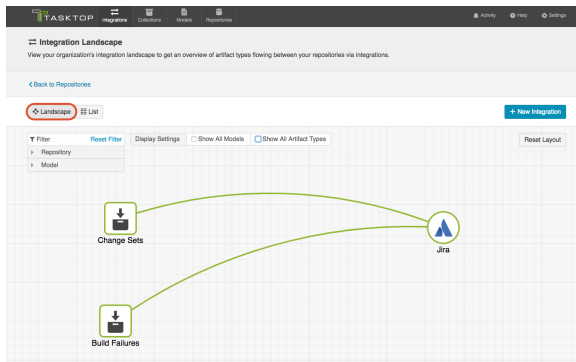
Click **Run** or **Stop** next to each integration you would like to update. You can also use the **Bulk Actions** button to run or stop all integrations.



## Viewing Your Integrations

See [Tasktop Editions table](#) to determine if your edition contains Integration Landscape View functionality.

When viewing your integrations, you have the option of viewing them in either Landscape or List mode.



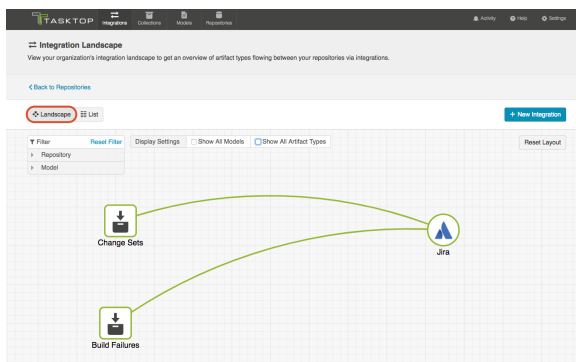
## Landscape View

See [Tasktop Editions table](#) to determine if your edition contains Integration Landscape View functionality.

Learn more about the Integration Landscape view in the video below:

Tasktop will default to the Landscape View, which enables you to visualize your entire integration landscape and see how your integrations relate to one another. Use our built-in filters to see as little or as much information as you'd like!

Here's a simplified view:



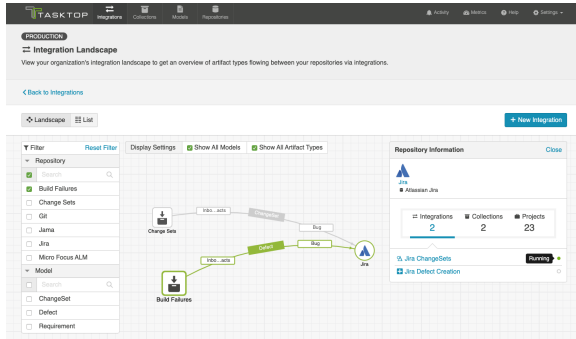
If you'd like to see additional information, you can utilize the filters, or click on a repository node to modify which information is shown.

Some examples of additional information you can see are:

- Models

- Artifact Types
- Artifact Creation Directionality Arrows
- List of all relevant integrations (see this by clicking on the repository node)
  - Indicator of whether each integration is running or not

Here's an example of a more detailed view:

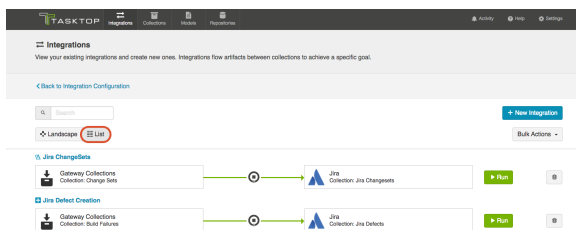


## List View

If you'd like, you can toggle to List view, which will show you a list of all integrations you have created.

You can use this view to:

- Start an Integration
- Stop an Integration
- Delete an Integration
- Click into an Integration and modify its configuration



## Tips and Tricks

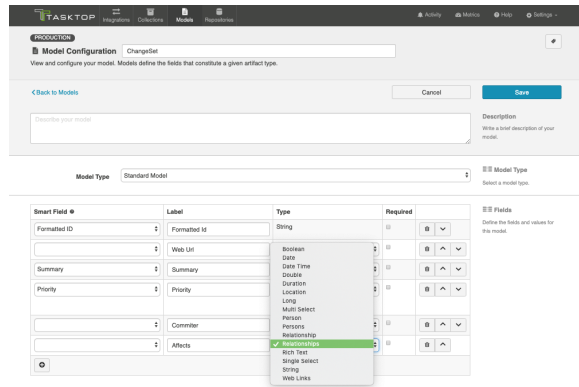
### Creating Relationships Between Newly Created Artifacts and Existing Artifacts

If you'd like to create relationships between your newly created artifacts and existing artifacts in the same repository, please follow the additional steps listed below:

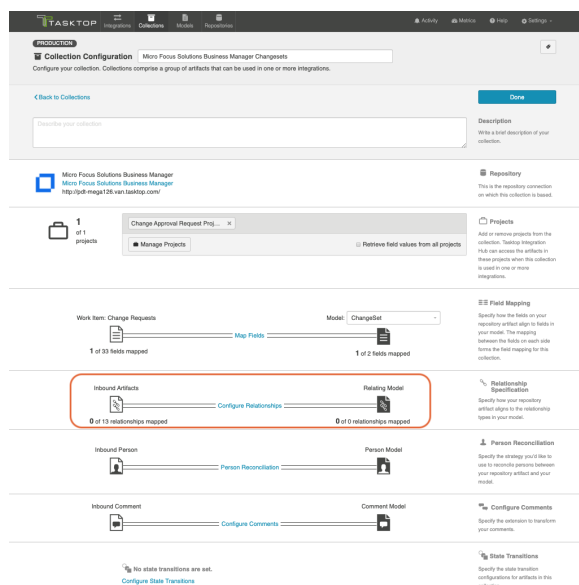
**At the Model level:** When creating your model, you can create a field that is of type “relationship” or “relationships”. You should use “relationship” when the newly-created artifact can only relate to one other artifact and “relationships” when the newly-created artifact can relate to multiple artifacts.

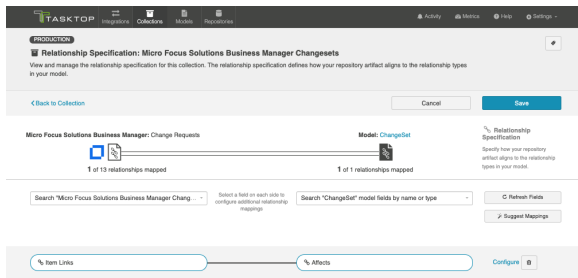
For example, the relationship field type, “Parent,” should generally be singular, as most artifacts usually only have a single parent. However, if the relationship field type is called “Blocks”, it can likely be plural, as one artifact can block many artifacts.

In the use case example described at the top of this page, I want the relationship to be “Affects” because any incoming changeset can affect many stories. So I’ll configure a *relationships* field.

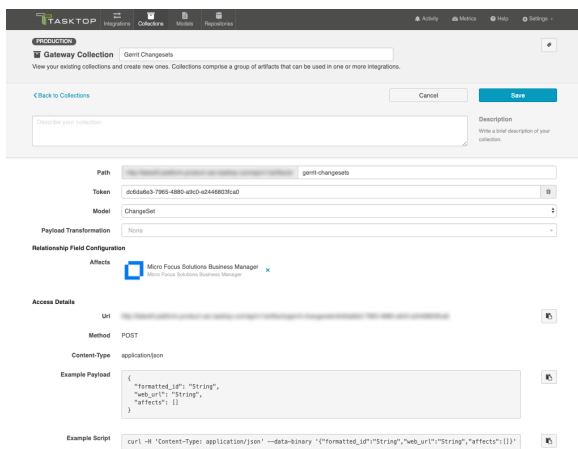


**At the Repository Collection level:** When creating your repository collection, you will need to map a field in your repository to the relationship(s) field in your model. So, in the same example, if you want the relationship between the new changeset and the existing story to be “affects”, but the relationship is actually called “items linked” in SBM, you would need to map those two fields. You’ll need to do this for each relationship type configured in your model.



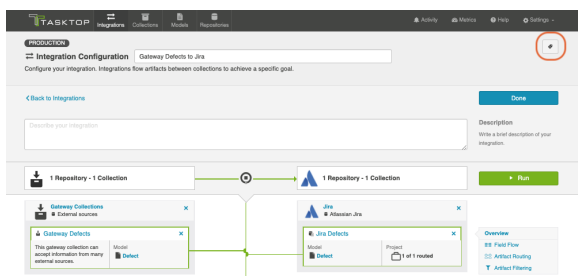


**At the Gateway Collection Level:** When creating your gateway collection, you will see that for each model field that is of relationship(s) type, you must specify the target repository that contains the related artifact(s). Once this is selected, the information needed for Tasktop to successfully locate the artifact will be added to the example payload.



## Viewing Associated Configuration Elements


To view associated configuration elements (such as collections or models that utilize the integration you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



 Associated Elements for Integration "Gateway Defects to Jira"

 1 Extension used by this Integration

- [Comment Extension](#)

 1 Gateway Collection used by this Integration

- [Gateway Defects](#)

 1 Model used by this Integration

- [Defect](#)

 1 Repository Collection used by this Integration

- [Jira Defects](#)

 1 Repository Connection used by this Integration

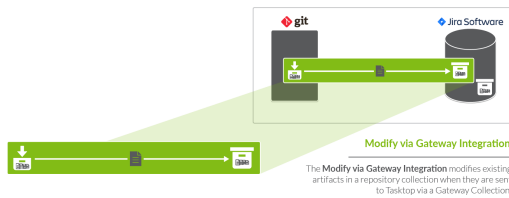
- [Jira](#)

Close

# Modify via Gateway

The *Modify via Gateway Integration* template is only available in Editions that contain the *Gateway add-on*. See [Tasktop Editions table](#) to determine if your edition contains this functionality.

## What is a Modify via Gateway Integration?



An *integration* is quite simply **the flow of information between two or more collections**. A *Modify via Gateway Integration*, specifically, locates and modifies existing artifacts in a work item or container collection that connects to a repository, when they are sent to Tasktop via a gateway collection. A gateway collection accesses event-based information in an external tool, such as Git or Jenkins, via an inbound webhook.

These types of events are “fire and forget” — they can modify something in your repository, but they don’t expect anything back. As such, they don’t mandate a full-blown two way synchronization; a lighter one-way integration can do the trick.

Here is an example of what you can do with the *Modify via Gateway* integration template:



### **A code commit updates a story:**

When a developer commits code in Git, Tasktop updates the Jira story with a link to the Git changeset.

When you configure a *Modify via Gateway* integration, you can customize the field flow and artifact filtering.

## Video Tutorial

Check out the video below to learn how to configure the *Modify via Gateway Integration* template.

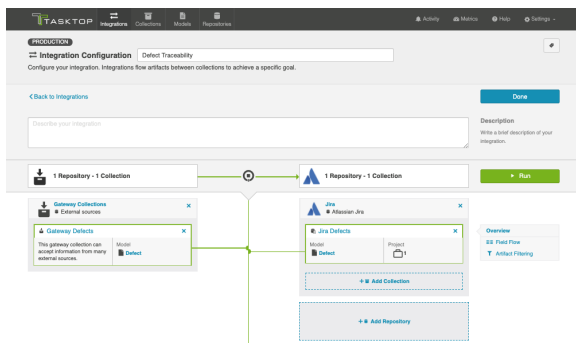


⚠️ This video assumes that you have already configured your repositories, models, and collections as outlined in the [Quick Start Guide](#).

## Use Case and Business Value

The **Modify via Gateway** integration creates traceability between artifacts across the software development lifecycle. Already existing artifacts in a repository collection will be located and modified in a specified way when artifacts are sent to Tasktop via a gateway collection.

For example, if your development team uses Gerrit for code review and Jira for its agile work management, but would like to know which defects in Jira a given code review affects, or conversely which code reviews are associated with a given defect, you could set up an integration that would find an already-existing defect in Jira anytime a code review is sent in and append one of its fields with that code review's URL. The integration can even include updating other Jira artifacts to which code reviews might pertain, such as stories and tech debt.



## Template Affordances

The Modify via Gateway Integration Template allows you to update already-existing artifacts in target work item (repository) or container (repository) collection when artifacts are sent to Tasktop via a gateway collection.



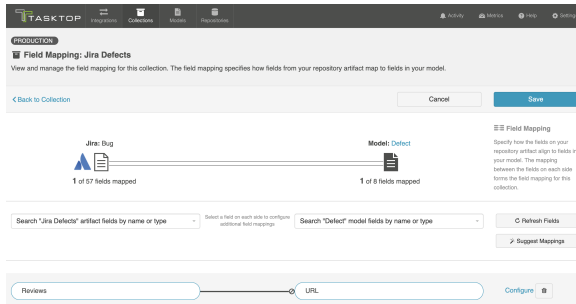
## Configuring a Modify via Gateway Integration

### Configuring your Repository Collection

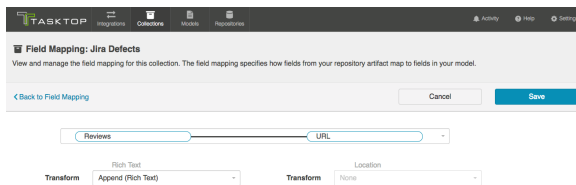
Before you begin configuring the integration itself, there are some steps that must be taken at the repository collection level:

To specify just how you would like incoming artifacts from your gateway collection to modify already existing artifacts in your repository collection, you need to identify which field(s) on your already-existing artifacts you would like to modify and then configure how the field(s) should be changed. In the example above, the URL to any incoming code reviews from a gateway collection is being added to the review field of the Jira defect.

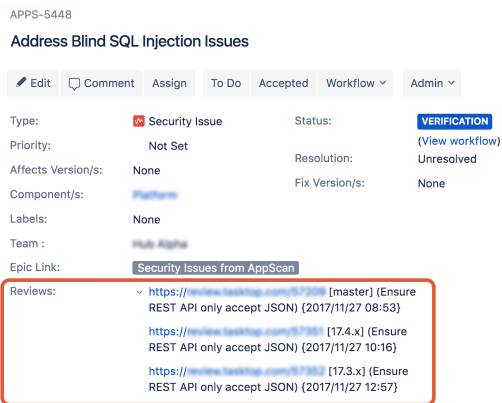
This means that the Jira collection-to-model mapping is configured as such:



And here are how the transformations are configured between these fields:

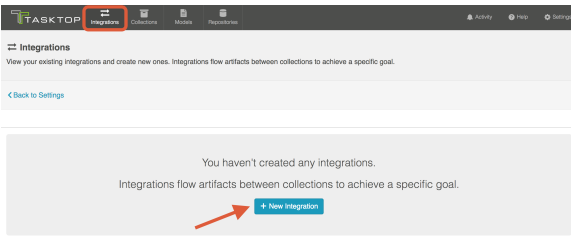


The **Append** transform means that new values will be added to the field value, rather than overwriting it, leaving the Jira artifact itself looking like this:



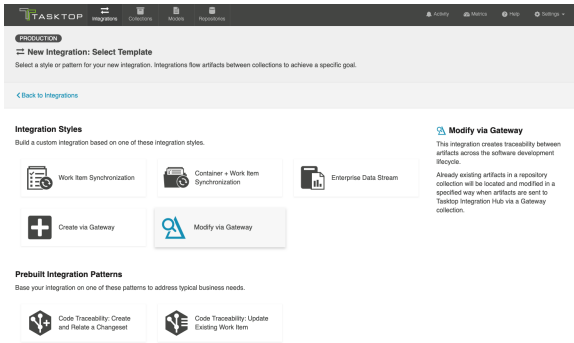
## Configuring Your Integration

To configure your integration, select **Integrations** at the top of the screen, then click **New Integration**.

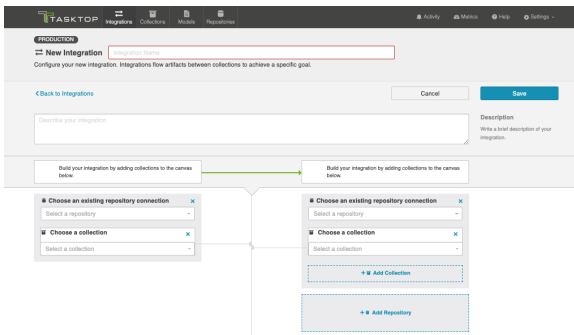


Select the **Modify via Gateway** template.

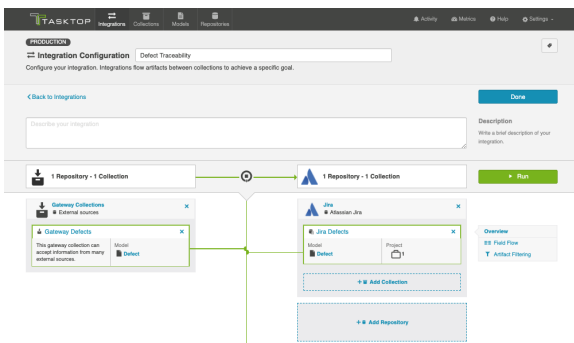
💡 Depending on the **edition** of Tasktop you are utilizing, you may not have all options shown here.



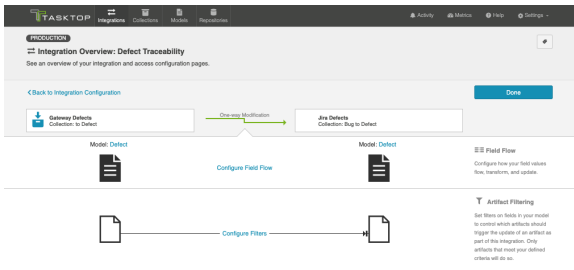
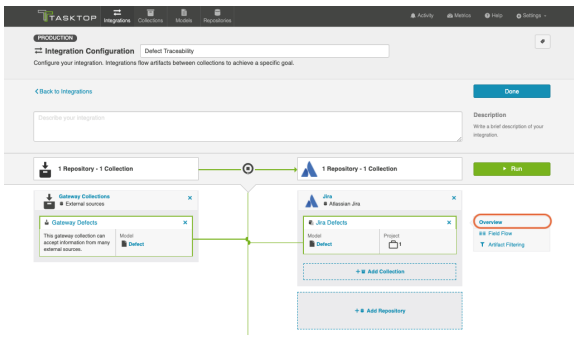
This will bring you to the New Integration screen:



Name your integration and select your repositories and collections:



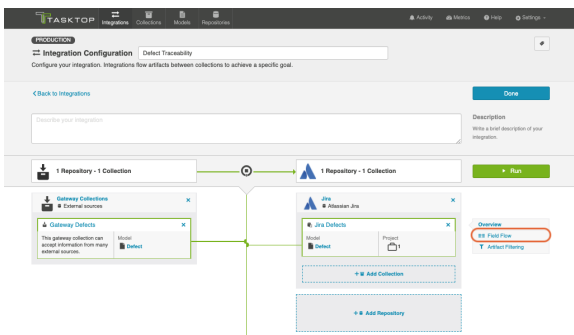
You can click the **Overview** link on the right side of the Integration Configuration screen to get to the main display page (shown in the second screen shot):



## Field Flow

The field flow configured for a given integration specifies which fields should flow in that integration. For Modify via Gateway integrations, you can choose to flow a given field (Update Normally) or to not flow a given field (No Update).

To get to the Field Flow screen, click **Field Flow** on the right side of the Integration Configuration screen.



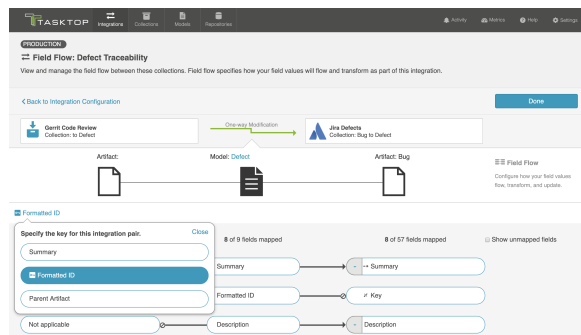
## Specifying Your Key

The first thing you will need to do when you get to the Field Flow screen is to specify your key.

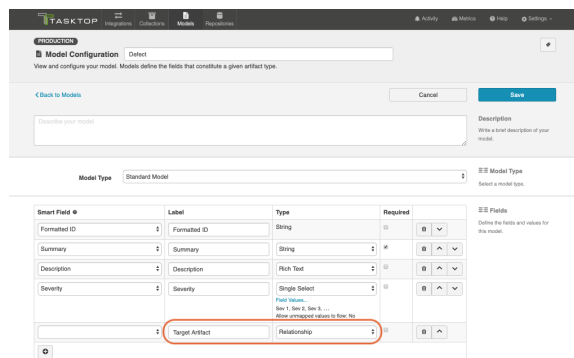
Specifying a key will enable Tasktop to find the existing artifact in your repository collection that is to be modified by the incoming gateway payload(s). The key can be a string or relationship field from the model.

If the key is a string field, then the value sent to that model field from the gateway payload will be used to look up the target artifact by Formatted ID. For this reason, the recommended field to use is the Formatted ID field.

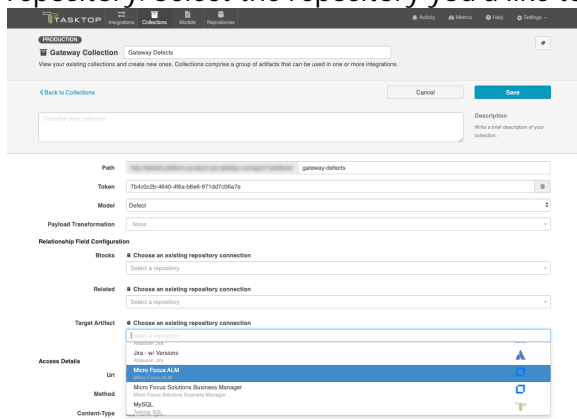
If the key is a relationship field, then the artifact it references in the gateway payload will be used as the target artifact.



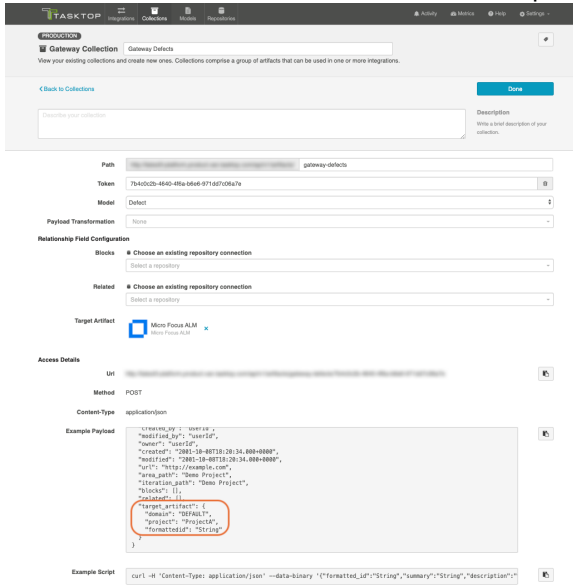
**Note:** Some repositories require extra information in order to uniquely identify a single artifact across multiple projects. One prime example is ALM. To ensure that enough information is sent in via your gateway collection to allow Tasktop to find the specific artifact you would like to modify, please take these steps:



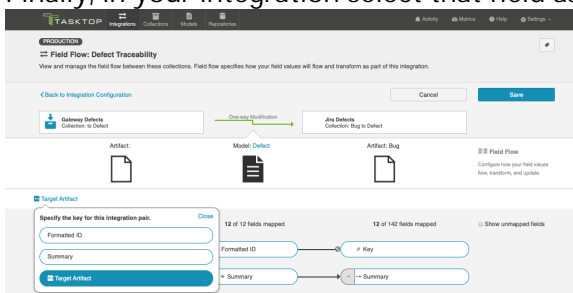
1. Add a field in your model of type relationship.
2. In your gateway collection, notice that for the new field you are prompted to pick a target repository. Select the repository you'd like to target in this gateway integration.



- When you save, note that the example payload will be updated to include the pieces of information we need for that field to uniquely find artifacts.



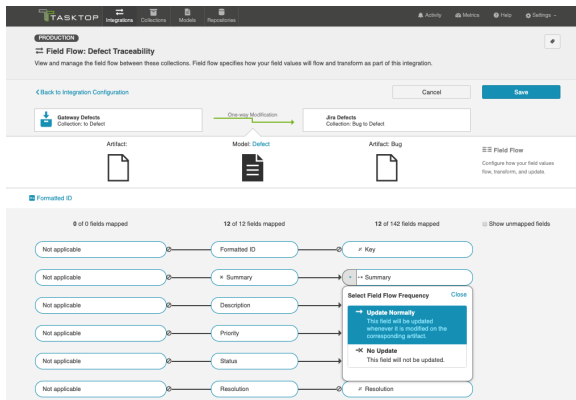
- Finally, in your integration select that field as your key on the Field Flow screen.



## Configure Field Flow

Once you have specified your key, you can configure your field flow. For each field, you can choose to flow information ('update normally') or not flow information ('no update'). You'll notice that field flow goes in one direction only – from the gateway collection *into* the repository or database collection.

You can see the names of the mapped artifact fields for each collection on the far left and far right, with the model fields displayed in the middle. By default, model fields without mapped repository fields are hidden. You can see all model fields by checking the **Show unmapped fields** checkbox. Constant values will be identified by a grey box and the constant value icon.



## Field Flow Icons

On the Field Flow page, you will see a number of icons, which will help you understand any special properties or requirements for each field. If you hover your mouse over an icon, you will see a pop-up explaining what the icon means. You can also review their meanings in the legend below:

Icon	Meaning
	<p>A constant value will be sent.</p> <p>Note that:</p> <ul style="list-style-type: none"> <li>If the icon is on the side of the collection, this means that a constant value will be sent to your model. This means that any time this collection is integrated with another collection, the <i>other</i> collection will receive this constant value for the field in question.</li> <li>If the icon is on the side of the model, this means a constant value will be sent to your collection. This means that any time this collection is integrated with another collection, that <i>this</i> collection will receive this constant value for the field in question.</li> </ul>
	Collection field is read-only and cannot receive data
	To create artifacts in your collection, this field must be mapped to your model.
	This is a required field in your model; it must be mapped to your collection.
	This field will not be updated as part of your integration, due to how you have configured it. This field flow configuration can be changed if you'd like.
	This field will not be updated as part of your integration because the mapping would be invalid. You do not have the option of changing this.
	This field will update normally as part of your synchronize integration; this means it will be updated whenever it is modified on the corresponding artifact.

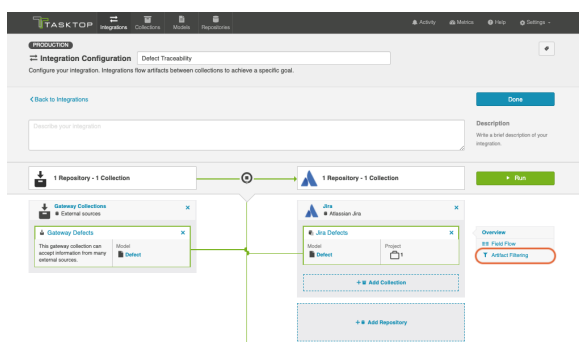
# Artifact Filtering

**Artifact Filtering** enables you to set filters on an integration in order to limit which artifacts are eligible to flow in your integration.

To use a field for artifact filtering, it must:

- Be a part of your model, and be mapped to the collection you are filtering from
- Be one of the following field types:
  - Single Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **only** fields that are already a part of the model will be considered for artifact filtering
  - Multi-Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **only** fields that are already a part of the model will be considered for artifact filtering
  - Date
  - Date/Time
  - Duration
  - String

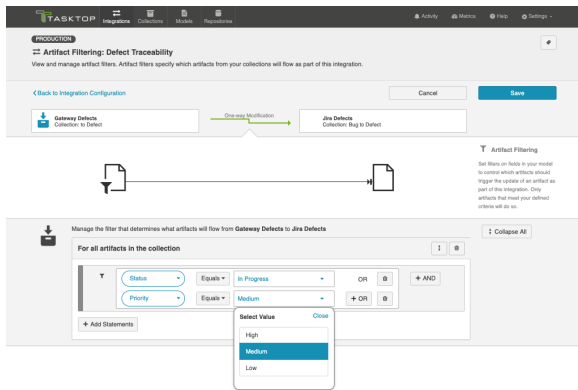
To configure Artifact Filtering, select **Create filters (optional)** from the Integration Configuration Overview screen, or select **Artifact Filtering** from the right pane of the Integration Configuration screen.



This will lead you to the Artifact Filtering Configuration screen, where you can configure one or more criteria for artifact filtering.

💡 You can click the **Collapse All** button to view an easy-to-read summary of your artifact filtering statements.





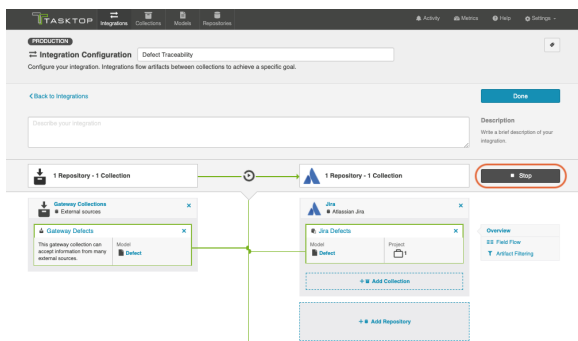
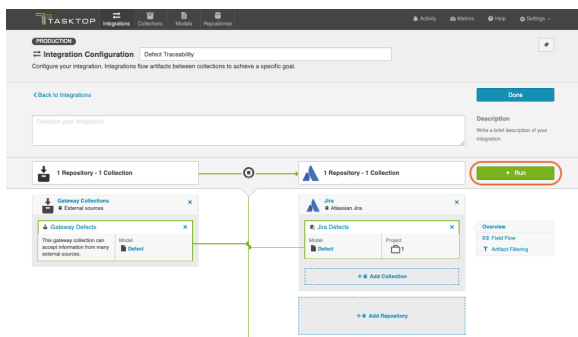
## Running your Integration

⚠ Please be aware that integrations will trigger changes in your end repositories and that misconfiguration can cause items to be duplicated or modified in unexpected ways. Additionally, there is no 'undo' in Tasktop or the repositories. If you have any questions, please [contact support](#).

There are two ways to start or stop your integration:

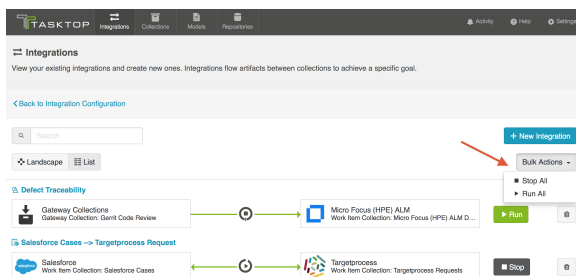
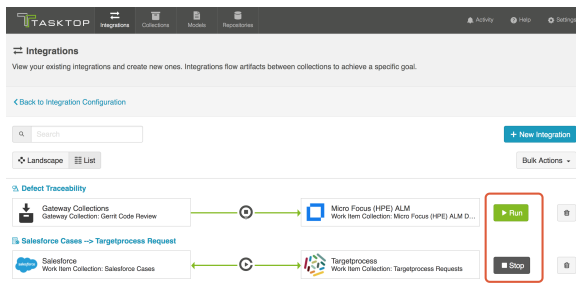
## From the Integration Configuration Screen

Simply click the **Run** button to run the integration, and the **Stop** button to stop the integration.



## From the Integrations List Screen

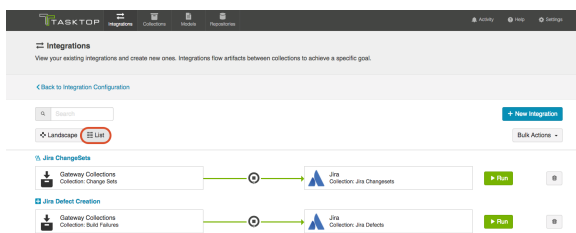
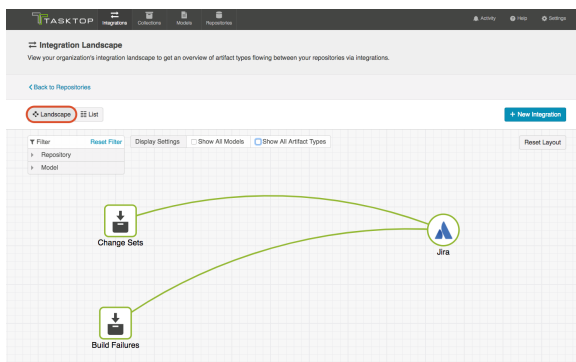
Click **Run** or **Stop** next to each integration you would like to update. You can also use the **Bulk Actions** button to run or stop all integrations.



## Viewing Your Integrations

See [Tasktop Editions table](#) to determine if your edition contains Integration Landscape View functionality.

When viewing your integrations, you have the option of viewing them in either Landscape or List mode.



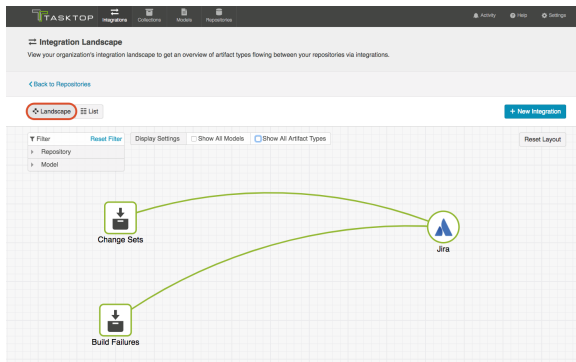
## Landscape View

See [Tasktop Editions table](#) to determine if your edition contains Integration Landscape View functionality.

Learn more about the Integration Landscape view in the video below:

Tasktop will default to the Landscape view, which enables you to visualize your entire integration landscape and see how your integrations relate to one another. Use our built-in filters to see as little or as much information as you'd like!

Here's a simplified view:

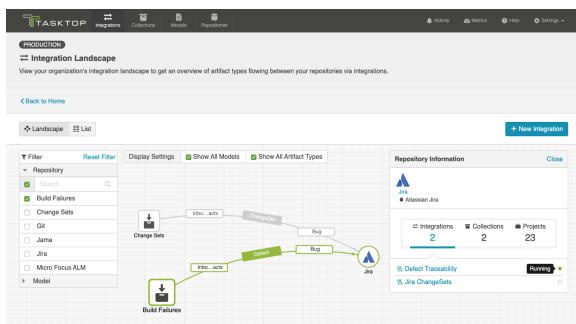


If you'd like to see additional information, you can utilize the filters, or click on a repository node to modify which information is shown.

Some examples of additional information you can see are:

- Models
- Artifact Types
- Artifact Creation Directionality Arrows
- List of all relevant integrations (see this by clicking on the repository node)
  - Indicator of whether each integration is running or not
- Collections
- Projects

Here's an example of a more detailed view:

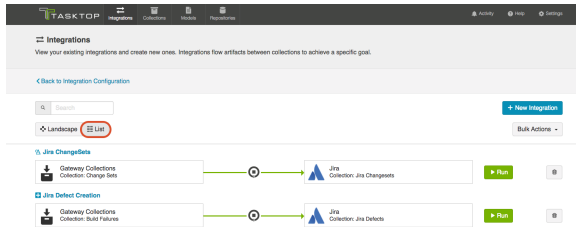


## List View

If you'd like, you can toggle to List view, which will show you a list of all integrations you have created.

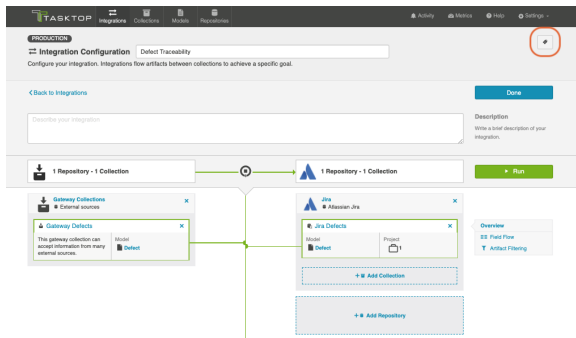
You can use this view to:

- Start an Integration
- Stop an Integration
- Delete an Integration
- Click into an Integration and modify its configuration



## Viewing Associated Configuration Elements

To view associated configuration elements (such as collections or models that utilize the integration you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



### Associated Elements for Integration "Defect Traceability"

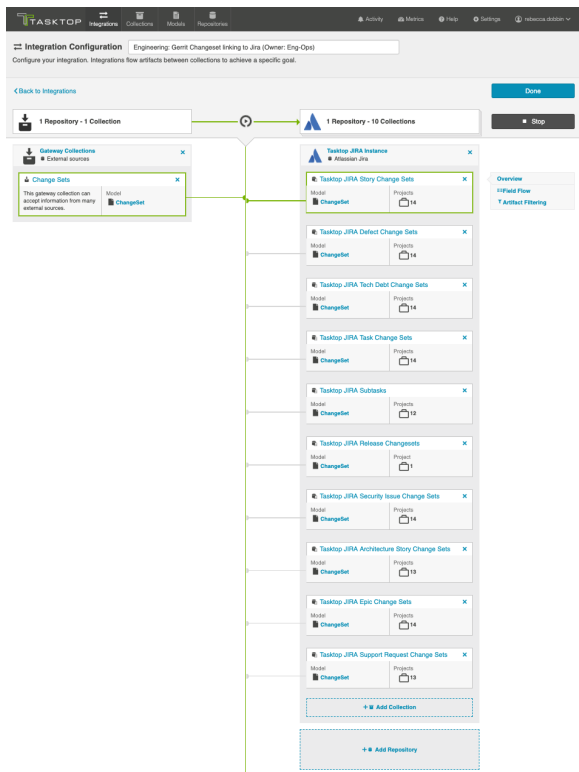
- 1 Extension used by this Integration
  - [Comment Extension](#)
- 1 Gateway Collection used by this Integration
  - [Gateway Defects](#)
- 1 Model used by this Integration
  - [Defect](#)
- 1 Repository Collection used by this Integration
  - [Jira Defects](#)
- 1 Repository Connection used by this Integration
  - [Jira](#)

Close

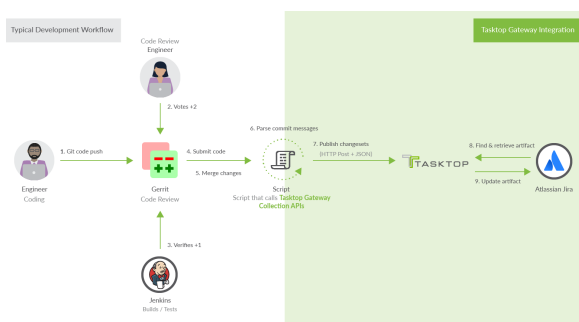
## Example Use Case

This is an example of how we at Tasktop utilize the Modify via Gateway template. Our integration flows changeset links and other information from Gerrit to a field on already-existing artifacts (such as stories, epics, and defects) in Jira.

Here's how the integration configuration screen looks for that integration:



The image below illustrates how the changeset is sent to Tasktop after the developers' normal workflow:



This is an example of the script that we use to automate the changesets being sent to Tasktop:

**Example Script**

```
#!/usr/bin/ruby

require 'rubygems'
require 'logger'
require 'net/http'
require 'openssl'
require 'json'
```

```

def getOption(name)
  return ARGV[ARGV.index("--"+name)+1]
end

def sendToLink(data)
  request = Net::HTTP::Post.new(LINK_URL)
  request.body = JSON.generate(data)
  request.content_type = 'application/json'
  request.basic_auth "tasktop", "tasktopSecret"
  uri = URI.parse(LINK_URL)
  response = Net::HTTP.start(uri.hostname, uri.port, :use_ssl => uri.scheme == 'https', :verify_mode =>
OpenSSL::SSL::VERIFY_NONE) do |http|
    http.request(request)
  end
  if ! response.kind_of? Net::HTTPSuccess
    LOGGER.warn "Error sending to link: #{response.body}"
  end
end

LINK_URL = "https://tt-data350:8443/api/v1/artifacts/changesets"
TASK_ID_PATTERN = /Task-Url:\s*https:\/\/\tasktop.atlassian.net\/browse\/([\^s]*)/
REVIEW_URL_PATTERN = /\.Reviewed-on:\s+([\^s]*)/m
LOGGER = Logger.new('/shared/gerrit/tasktop-site/logs/hook-change-merged.log', 'monthly')
ENABLED_PROJECT_KEYS = ["APPS", "SYN", "SDK", "PLAT", "OPS", "CON", "DEV", "QA", "RLIASE"]

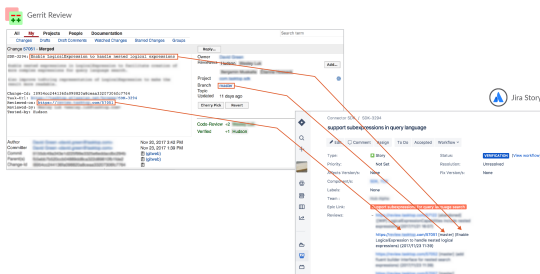
project = getOption('project')
commit = getOption('commit')
branch = getOption('branch')

LOGGER.debug("Processing merge for commit #{commit} on project #{project}")

gitPath = ENV['GIT_DIR']
message = `git --git-dir #{gitPath} show -s --format=%B #{commit}`
taskIdMatch = TASK_ID_PATTERN.match(message)
if taskIdMatch
  taskKey = taskIdMatch.captures[0]
  LOGGER.debug("Detected taskKey: #{taskKey}")
  taskKeyMatches = ENABLED_PROJECT_KEYS.any? { |project| taskKey.start_with?(project + "-")}
  if ! taskKeyMatches
    LOGGER.info("#{taskKey} project not enabled, skipping");
    exit()
  end
  reviewUrlMatch = REVIEW_URL_PATTERN.match(message)
  webUrl = nil
  if reviewUrlMatch
    webUrl = reviewUrlMatch.captures[0]
  else
    LOGGER.error("Could not get webUrl from commit #{commit}")
    webUrl = "commit #{commit}"
  end
  firstLineOfMessage = message.lines.first.chomp
  firstLineOfMessage = firstLineOfMessage.gsub(/#{taskKey}? /, '')
  sendToLink({"formatted_id" => taskKey, "info" => "#{webUrl} [#{branch}] (#{firstLineOfMessage})"})
else
  LOGGER.debug("No task key found")
end
end

```

This image more clearly highlights how these changesets are reflected on the Jira artifacts:

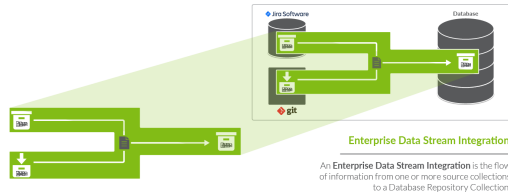




# Enterprise Data Stream

The Enterprise Data Stream Template is only available in Editions that contain the Enterprise Data Stream add-on. See [Tasktop Editions table](#) to determine if your edition contains this functionality.

## What is an Enterprise Data Stream Integration?



An *integration* is quite simply **the flow of information between two or more collections**. An Enterprise Data Stream Integration, specifically, is the flow of information from one or more source collections (either Work Item (Repository) Collections, Container (Repository) Collections, or Gateway Collections) to one central table held in a Work Item (Database) Collection.

When you configure your Enterprise Data Stream Integration, you can customize the field flow, artifact routing, and artifact filtering.

## Video Tutorial

Check out the video below to learn how to configure an Enterprise Data Stream Integration.

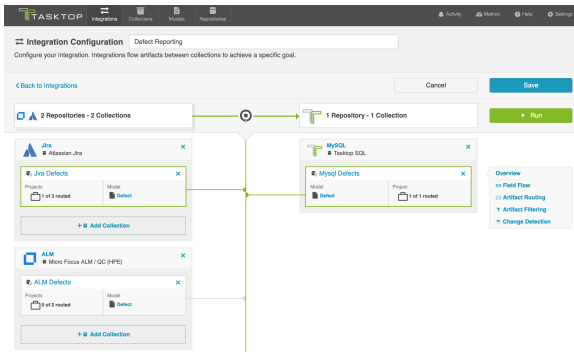
⚠ This video assumes that you have already configured your repositories, models, and collections as outlined in the [Quick Start Guide](#).

## Use Case and Business Value

This integration simplifies enterprise reporting by unlocking software lifecycle data from its application tool silos and providing a rich data repository for near real-time analytics. Records will be created in a single database when artifacts from one or more collections are created or changed.

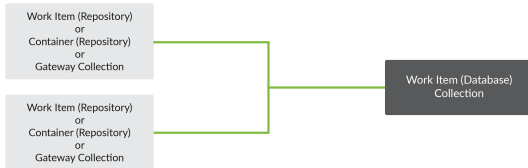
For example, if your organization uses multiple tools for defect discovery and resolution, such as Atlassian Jira and ALM, but would like to report on defects across both of the tools, you could set up an integration that would flow artifacts from your Jira and ALM collections into a single database table. You could then report directly from this aggregated table or, more likely, ETL it into your existing reporting infrastructure.





## Template Affordances

The Enterprise Data Stream template allows you to flow artifacts from multiple repository collections and/or gateway collections into a single database collection.



*Gateway Collections are only available in editions that contain the Gateway add-on. See [Tasktop Editions table](#) to determine if your edition has this functionality.*

## Key Concepts

Before you begin, there are a few concepts it's important to understand when configuring an Enterprise Data Stream Integration.

### Data Structures

An Enterprise Data Stream Integration populates a table with rows corresponding to the state of artifacts at a specific point in time. As an artifact changes, new rows are inserted corresponding to the new state of the artifact. The result is that each artifact has a series of rows corresponding to the state of the artifact at each point in time. The rows for all artifacts in a table can be thought of as an event stream.

**Note:** Tasktop will examine your repositories for changes as specified in the [change detection interval](#) that you have configured. This means that if you have configured the change detection interval to be 1 minute, and a given artifact is changed twice in that minute, you'll only get a single record that reflects both changes.

The database table populated by the Enterprise Data Stream Integration has columns corresponding to fields in the artifact model, as well as some built-in fields that are designed to facilitate reporting. The following is an example of a database table corresponding to a simple defect model:

```

CREATE TABLE `Defect` (
  `id` BIGINT (19) AUTO_INCREMENT,
  `formatted_id` VARCHAR (1000) NOT NULL,
  `project` VARCHAR (255) NOT NULL,
  `type` VARCHAR (255) NOT NULL,
  `severity` VARCHAR (255) NOT NULL,
  `status` VARCHAR (255) NOT NULL,
  `summary` VARCHAR (1000) NOT NULL,
  `repository_id` VARCHAR (255),
  `repository_url` VARCHAR (255),
  `artifact_id` VARCHAR (255),
  `artifact_url` VARCHAR (255),
  `artifact_event_type` VARCHAR (255),
  PRIMARY KEY (`id`)
);

```

## Database Output

### Default Information that Tasktop will Flow

The following columns represent information that will automatically be flowed to your database table.

Column	Description
id*	A surrogate key, can be used in reports to uniquely identify a row.
repository_id*	The unique identifier of the connection, can be used in reports to identify a repository connection.
repository_url*	The URL of the repository, can be used in reports to identify a repository.
artifact_id*	An ID of an artifact that is globally unique, can be used in reports to uniquely identify an artifact across repositories and collections. The value of the <code>artifact_id</code> is an opaque value; assumptions should not be made about its structure or content. It should be noted that the <code>artifact_id</code> does not correspond to the id of the artifact as it is represented in the repository itself, but is useful for reporting since it is globally unique.
artifact_url	The URL of the artifact for browser access, can be used in reports to identify an artifact.
artifact_event_type	The type of event for the artifact that caused this entry. It can be used to see if the artifact has been added, changed or removed from the collection.

\*Denotes that this is a required field, meaning that your target database table will need to have a column to store this information.

**Note:** If you use the Suggest DDL to create your table, all of the fields above will be included. If you are creating your table without that mechanism, you'll need to ensure that a column exists for the required pieces of information and, ideally, for the non-required fields as well. Your database table columns will need to be named as displayed above in either upper or lower case, but with the underscores as displayed.

## Ordering of Rows

Though it may appear that rows in the table are inserted an order corresponding to the point in time that changes occurred, the order of rows in the table is not guaranteed. Reports should use a mapped field from the model (such as `modified`) to determine when a change occurred.

## Artifact Event Type

In the artifact event type column of your database table, you'll see either "changed", "removed", or "filtered."

### Changed

Changed indicates that either an existing artifact was changed or that a new artifact was added to your collection.

### Removed

Removed indicates that a given artifact is in a project that has been removed from the collection. Here is a sample scenario to illustrate this event type:

In this Enterprise Data Stream Integration Project B and C are routed to the database table in my SQL collection at the start of an integration. Artifacts flow and records get written out:

id	formatted_id	project	type	created	modified	severity	status	summary	description	repository_id	repository_uri	artifact_id	artifact_uri	artifact_event_type
1	TPB-B	Test...	Bug	2016-...	2016-0...	Blocker	To Do	d33269d5e...	desc	c00480c-6...	http://ga-jira...	room.ta...	http://ga-j...	changed
2	TPB-1	Test...	Bug	2015-...	2016-0...	Major	To Do	test bug B1	test bug B	c00480c-6...	http://ga-jira...	room.ta...	http://ga-j...	changed
3	TPC-1	Test...	Bug	2015-...	2016-0...	Major	To Do	test bug C1	test bug C	c00480c-6...	http://ga-jira...	room.ta...	http://ga-j...	changed

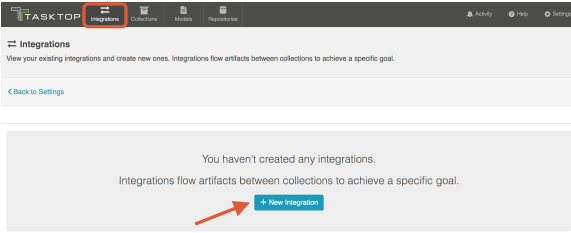
Project C is then removed from the source collection. At next full scan (one of the [change detection intervals configured on the General \(Settings\) screen](#)), you'll see an event to denote that any artifacts in that collection have been removed:

id	formatted_id	project	type	created	modified	severity	status	summary	description	repository_id	repository_uri	artifact_id	artifact_uri	artifact_event_type
1	TPB-B	Test...	Bug	2016-...	2016-0...	Blocker	To Do	d33269d5e...	desc	c00480c-6...	http://ga-jira...	room.ta...	http://ga-j...	changed
2	TPB-1	Test...	Bug	2015-...	2016-0...	Major	To Do	test bug B1	test bug B	c00480c-6...	http://ga-jira...	room.ta...	http://ga-j...	changed
3	TPC-1	Test...	Bug	2015-...	2016-0...	Major	To Do	test bug C1	test bug C	c00480c-6...	http://ga-jira...	room.ta...	http://ga-j...	changed
4	TPC-1	Test...	Bug	2015-...	2016-0...	Major	To Do	test bug C1	test bug C	c00480c-6...	http://ga-jira...	room.ta...	http://ga-j...	removed

**Note:** If the project is added back to the collection and routed, records will not instantly be written out for all artifacts in that project; this will happen only when those artifacts change again.

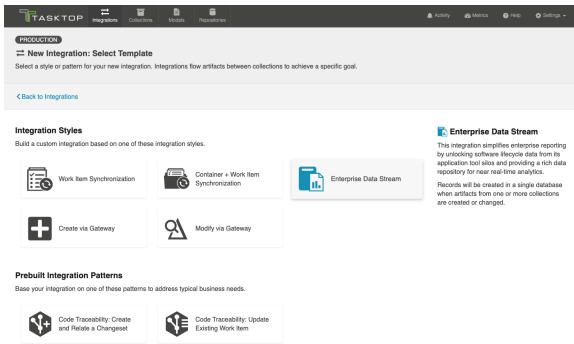
## Configuring an Enterprise Data Stream Integration

To configure your integration, select **Integrations** at the top of the screen, then click **New Integration**.

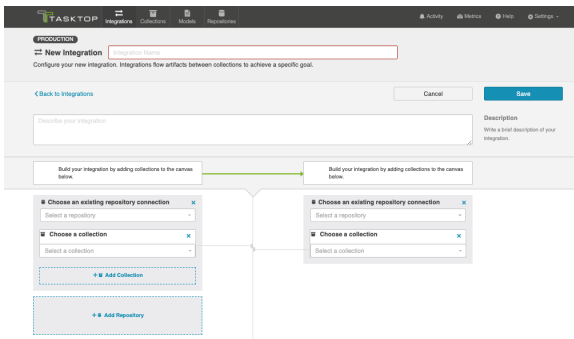


Select the **Enterprise Data Stream** template.

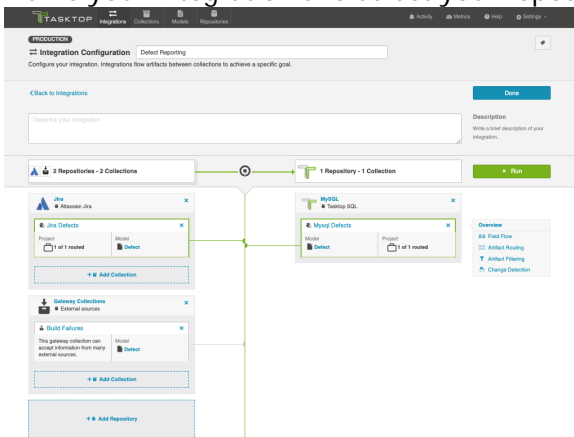
💡 Depending on the **edition** of Tasktop you are utilizing, you may not see all options shown below.



This will bring you to the New Integration screen:

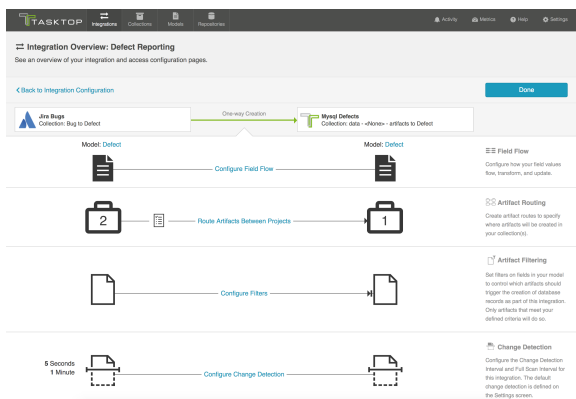
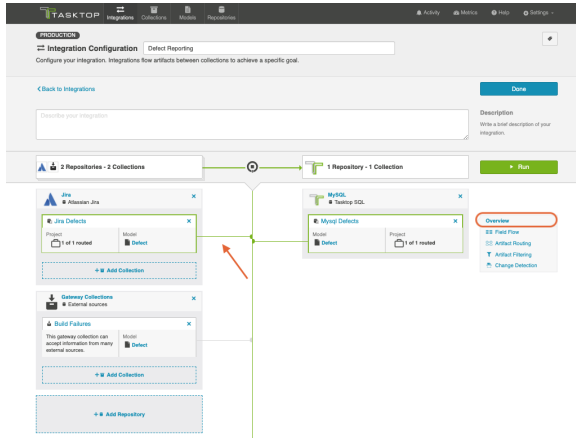


Name your integration and select your repositories and collections.



You can click the **Overview** link on the right side of the Integration Configuration screen to get to the main display screen (shown in the second screenshot).

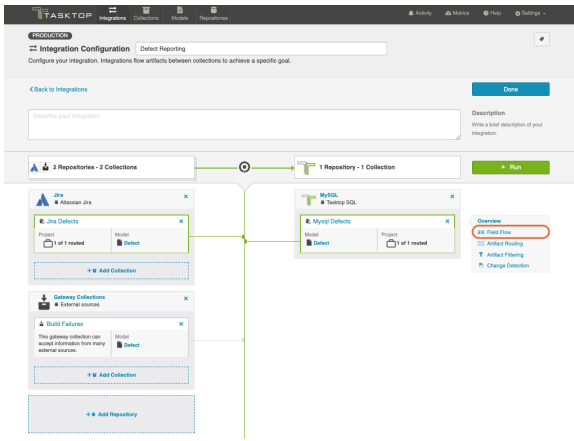
**Note:** The Overview screen will only show two repositories at a time — one source repository and one target repository. If there are multiple source repositories in your integration, make sure the one you are interested in is selected before clicking **Overview**.



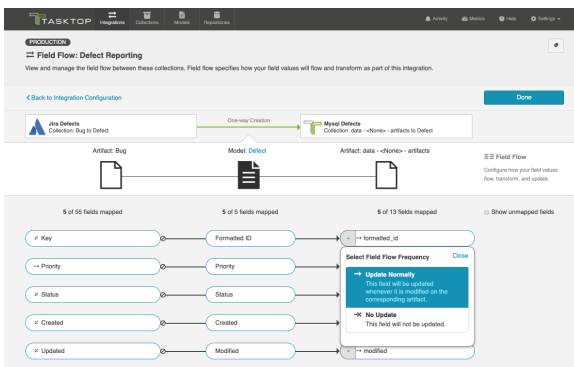
## Field Flow

The field flow configured for a given integration specifies which fields should flow in that integration. For Enterprise Data Stream integrations, you can choose to flow a given field (Update Normally) or to not flow a given field (No Update).

To view field flow, select the source repository you are interested in (you will see it highlighted in green once selected), and then click **Field Flow**.



You will be directed to the Field Flow screen.



You can choose to flow a field ('update normally') or not flow it ('no update'). You'll notice that field flow goes in one direction only — from the repository or gateway collection *into* the database collection.

You can see the names of the mapped artifact fields for each collection on the far left and far right, with the model fields displayed in the middle. By default, model fields without mapped repository fields are hidden. You can see all model fields by checking the **Show unmapped fields** checkbox. Constant values will be identified by a grey box and the constant value icon.

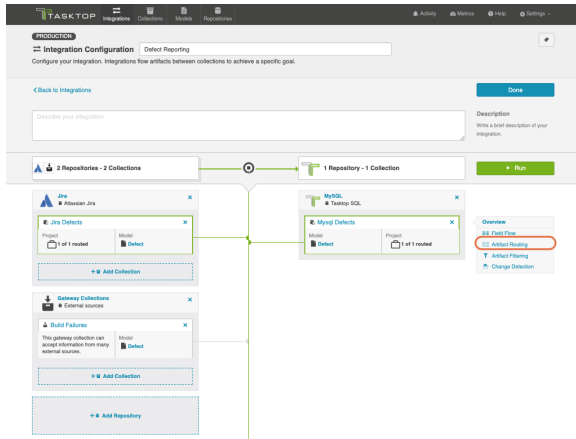
**Note:** The field flow settings behave a bit differently for Constant Values. This is because constant values exist as part of your Tasktop configuration, and not on the artifact itself. Therefore, changes in constant values are not detected in the same way that updates made on the actual artifact are detected. If you change the constant value that is linked to your model, your integration will not automatically detect this update and sync it over. The new value will only flow if another field on that artifact is updated.

## Artifact Routing

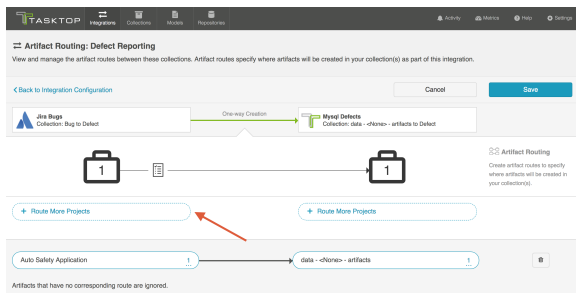
For an Enterprise Data Stream Integration, Artifact Routing is used to specify which projects (or other containers) you would like to participate in your integration. For example, your Jira Bugs collection

may contain 10 different projects which are utilized in various integrations. However, for the purpose of your Enterprise Data Stream Integration, you may want only one of those projects to participate. You can specify that project on the Artifact Routing Screen.

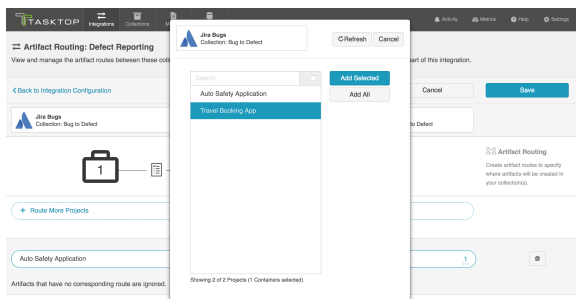
To configure Artifact Routing, select the relevant repositories and then click **Artifact Routing**.



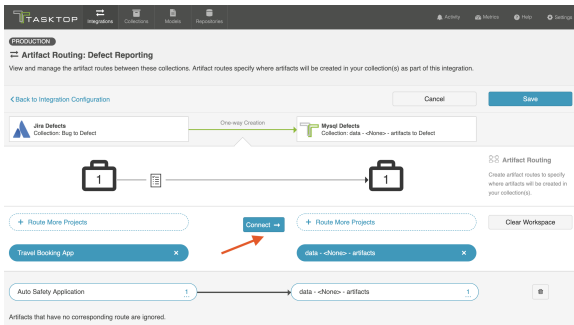
This will bring you to the Artifact Routing screen. You can click **Route More Projects** to add additional projects to your route.



Select the projects you would like to participate in the integration and click **Add Selected**.

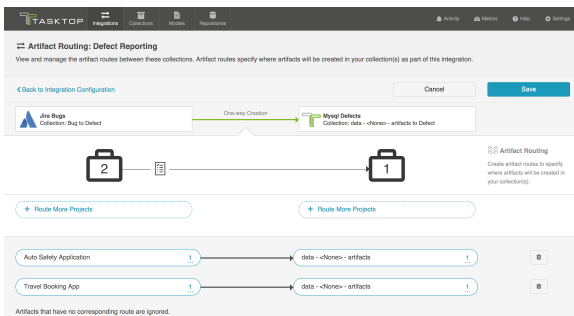


Click **Connect**.



You will see your artifact route on the pane below. Click **Save** and **Done**.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.



## Artifact Filtering

When configuring your integration, you have several options available to refine which artifacts are eligible to flow. The final mechanism available is *artifact filtering*, which is configured at the Integration level. Artifact Filtering allows you to filter which artifacts flow in your integration, based on a field value on that artifact.

To use a field for artifact filtering, it must:

- Be a part of your model, and be mapped to the collection you are filtering from
- Be one of the following field types:
  - Single Select
    - Note that in cases where **allow unmapped values to flow** is enabled in the model, **only** fields that are already a part of the model will be considered for artifact filtering
  - Multi-Select
    - Note that in cases where **allow unmapped values to flow** is enabled in the model, **only** fields that are already a part of the model will be considered for artifact filtering
  - Date
  - Date/Time
  - Duration
  - String

**Tip:** Note that you can utilize our transforms to filter based on an 'unsupported' collection field type, if that field is mapped to a supported field type in your model. For example, you could filter based on a Boolean field in your repository, if that boolean field is mapped to a single select field in your model.



# Unique Behavior for Enterprise Data Stream

The filtering behavior is somewhat unique when using the Enterprise Data Stream Template:

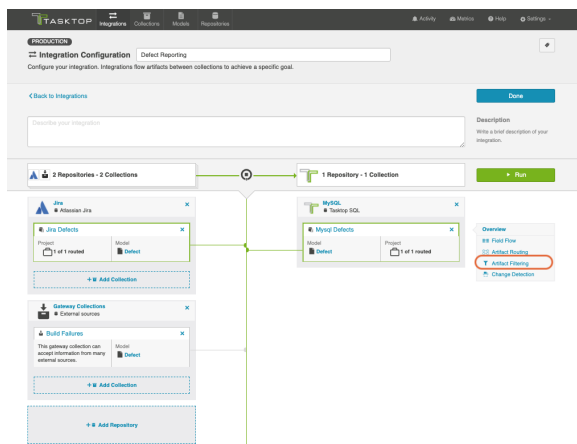
Though setting filters is meant to limit which artifacts flow in an integration, the impacts of setting filters on an Enterprise Data Stream Template are somewhat unique. Because it would not be ideal to have records in your database output that represent artifacts that have been filtered in an integration, given that these records would be stale and would not denote why a given artifact was not changing over time, it is the case that artifacts that are filtered on an Enterprise Data Stream Integration will still have records written out to the database but will have the "filtered" event type denoted.

Note the following:

- When you set a filter on an Enterprise Data Stream integration, records will not automatically be written out for artifacts that do not meet filtering criteria. When artifacts that should be filtered out change, we'll then write out a record with the "filtered" event type.
- When a once filtered artifact field changes such that it now meets the filter criteria set, records will be written out right away.
- If you relax the filter and more artifacts are now in scope, the now in scope artifacts will only flow when the artifacts themselves change again.
- If an artifact is filtered out of the Enterprise Data Stream Integration, and then its project is removed from the collection, records will be written out for all artifacts in that collection at next full scan and marked as "removed", whether or not they have been filtered out of the integration (This effectively means that the "removed" designation supersedes "filtered" designation.)
  - If you add the project back to the collection and routed in the integration, changes to artifacts will create a new record with either the "changed" or "filtered" event type, depending on whether or not the artifact meets the filter criteria.

## Configuring Artifact Filtering

To configure Artifact Filtering, select the relevant repository, then click **Artifact Filtering** from the right pane of the Integration Configuration screen.

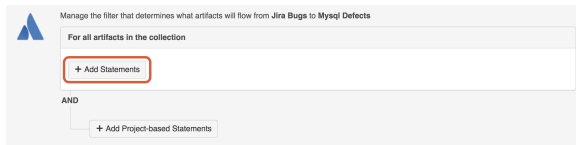


This will lead you to the Artifact Filtering Configuration screen, where you can configure your artifact filtering statement(s).

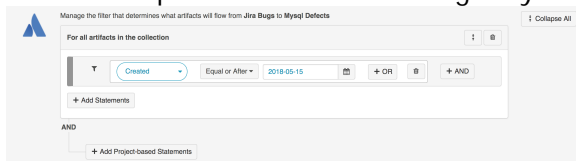
You can either add a statement that will apply to **all artifacts in your collection**, or to **all artifacts within certain projects of your collection**.

## Apply Filter to All Artifacts in Collection

To apply a filter to all artifacts in the collection, simply click **+ Add Statements**.

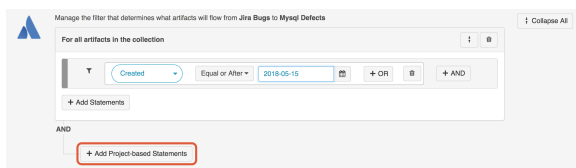


Use the drop-down menus to configure your filter fields and values:

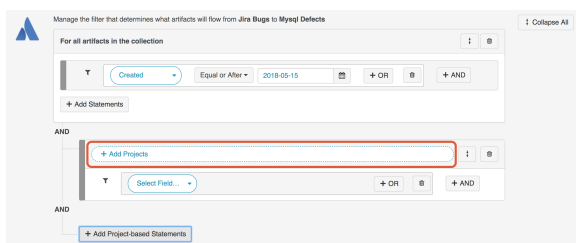


## Apply Filter to Artifacts in Certain Projects

To apply a filter to artifacts within a specific project, click **+ Add Project-based Statements**.

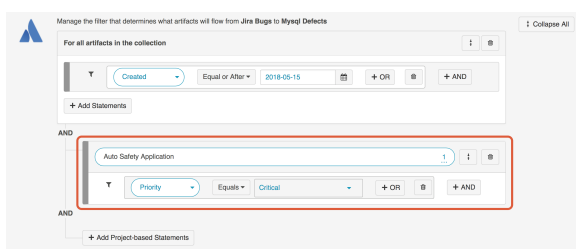


Click **+ Add Projects** to select your project.



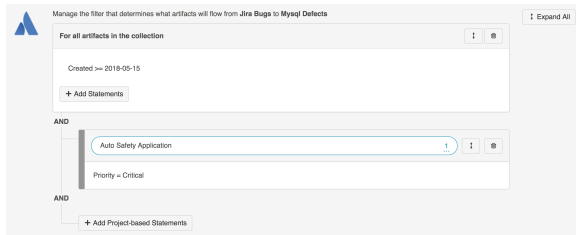
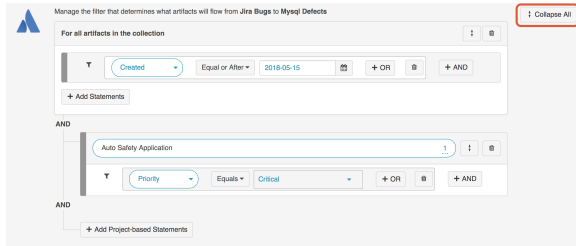
Select the project(s) you'd like your filter to apply to.

Then click **Select Field...** to begin configuring your filtering statement.



## Viewing Artifact Filter Statements

You can click the **Collapse All** button to view an easier-to-read version of your artifact filtering statements.



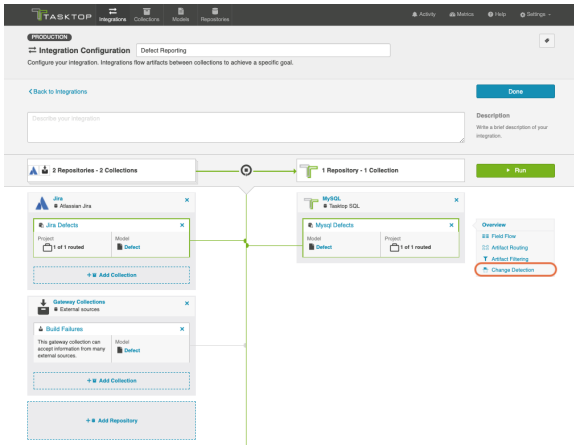
## Change Detection

Tasktop's default global change detection settings can be found on the [General \(Settings\)](#) screen. However, if you'd like to override the global defaults, you can configure integration-specific change detection and full scan intervals by clicking the **Change Detection** link.

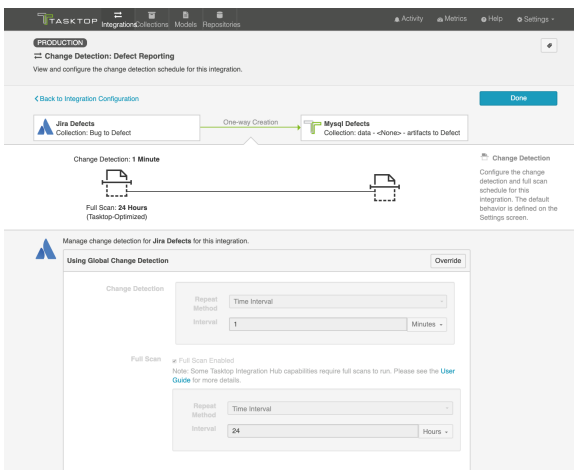
The **Change Detection Interval** is the time between polling requests to detect *only changed artifacts*. This defaults to 1 minute on the General (Settings) screen, but can be customized as desired.

The **Full Scan Interval** is the time between polling requests to detect changed artifacts, in which *all* artifacts of a collection are scanned. Not all changes to an artifact will register as a change. Some repositories do not mark items as changed when (for example) a relationship is added or an attachment has changed. These may not be picked up by regular Change Detection, but will be picked up by a Full Scan. This defaults to 24 hours on the General (Settings) screen, but can be customized as desired.

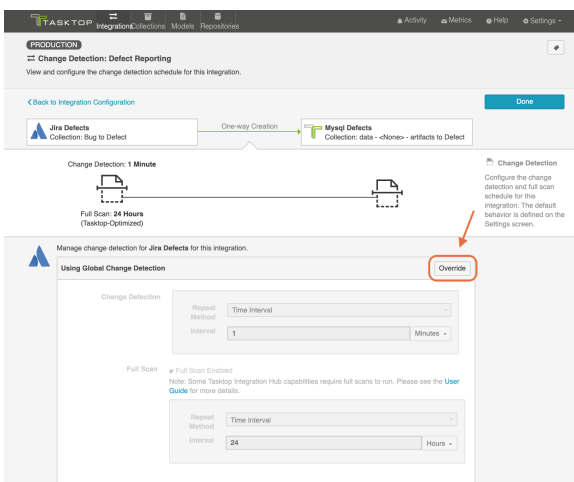
To configure integration-specific change detection, select the relevant repository, then click the **Change Detection** link.



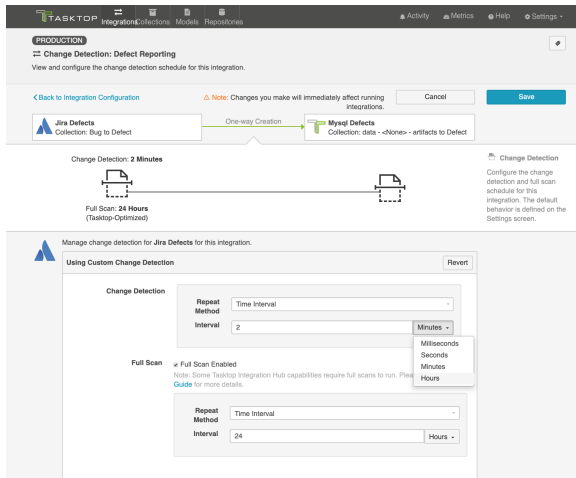
This will bring you to the Change Detection screen, where you can view the current change detection and full scan intervals configured for the selected collection in this integration. These will default to the global intervals configured on the General (Settings) screen.



To override the current settings, click the **Override** button. This will allow you to set a custom change detection and/or full scan interval for the collection within the context of this integration. Note that these custom settings will only impact *this* integration; they will not impact other integrations that make use of the same collections.



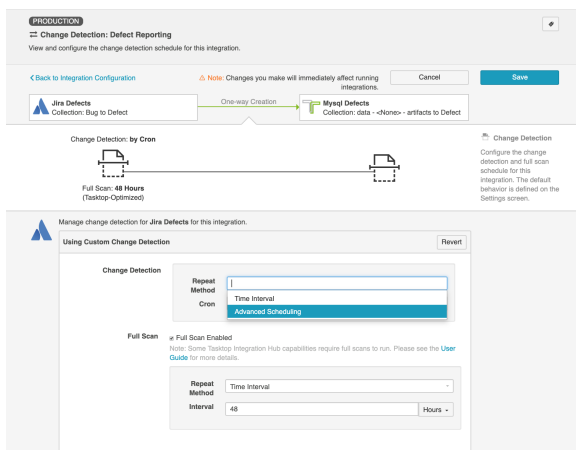
Once you click **Override**, you will be able to configure custom change detection and full scan intervals for the collection within the context of this integration.



In addition to customizing change detection and full scan intervals, you can also schedule change detection or full scan using cron expression.

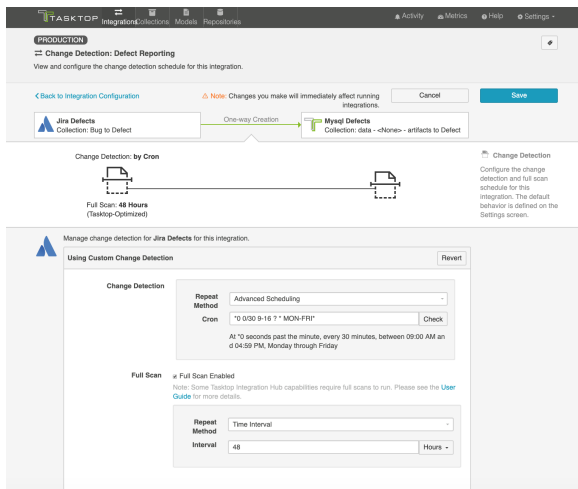
Using cron expression, you can configure complex schedules by running change detection or full scan during certain hours of the day, certain days of the week, or specific days of the month. Learn more in our FAQ [here](#).

To utilize cron expression for your full scan or change detection, select **Advanced Scheduling** in the Repeat Method field.

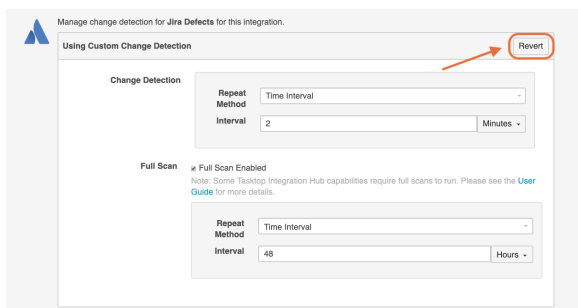


You can then enter the cron expression for your desired scheduling. For example, if you would like to run a full scan every 30 minutes from 9am to 5pm from Monday through Friday, it would be written as follows: `*0 0/30 9-16 ? * MON-FRI*`

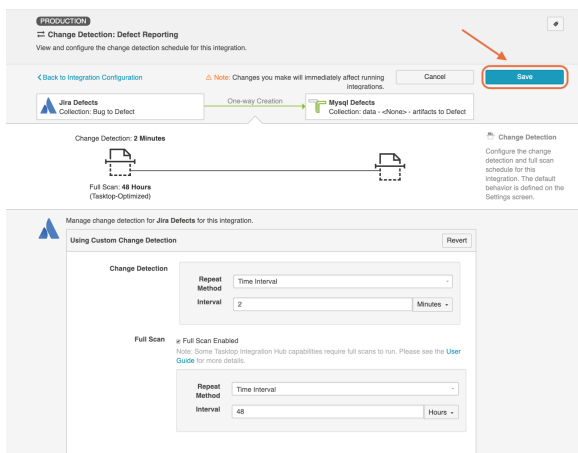
To ensure that your cron expression is valid, click the **Check** button. If valid, a readable form of the cron expression will be displayed. If the cron expression is invalid, an error message will appear.



If you'd like to restore the global change detection settings, simply click the **Revert** button to remove the custom settings.

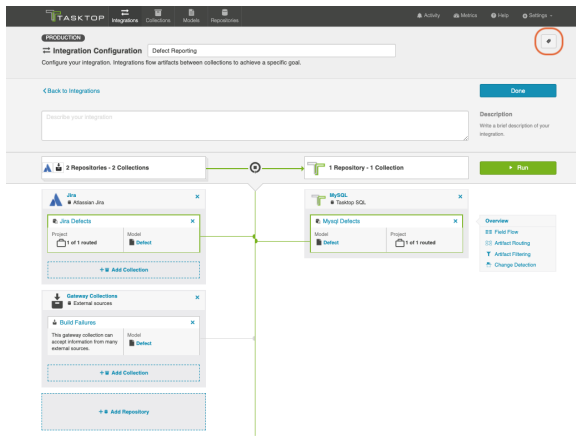


Once you've updated the change detection settings as desired, click **Save** and **Done** to save your changes.



## Viewing Associated Configuration Elements

To view associated configuration elements (such as collections or models that utilize the integration you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



### Associated Elements for Integration "Defect Reporting"

- 2 **Repository Collections used by this Integration**
  - [Jira Defects](#)
  - [Mysql Defects](#)
- 1 **Gateway Collection used by this Integration**
  - [Build Failures](#)
- 1 **Model used by this Integration**
  - [Defect](#)
- 2 **Repository Connections used by this Integration**
  - [Jira](#)
  - [MySQL](#)
- 1 **Extension used by this Integration**
  - [Comment Extension](#)

Close

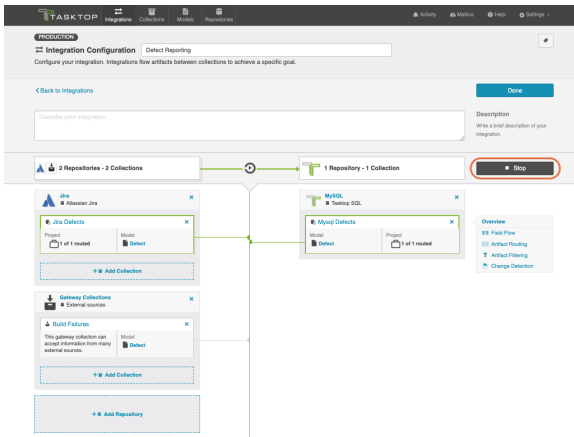
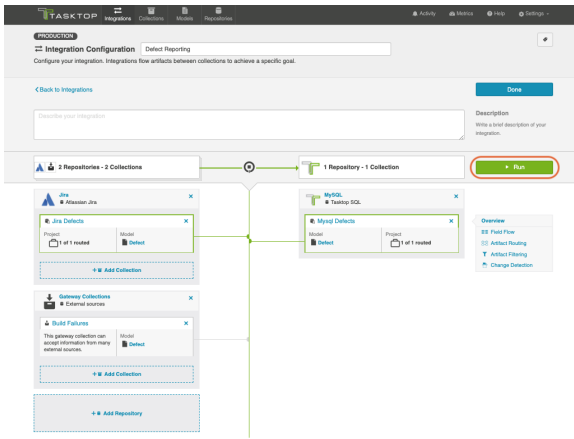
## Running your Integration

**⚠** Please be aware that integrations will trigger changes in your database and that misconfiguration can cause items to be duplicated or updated in unexpected ways. Additionally, there is no 'undo' in Tasktop or the database. If you have any questions, please [contact support](#).

There are two ways to start or stop your integration:

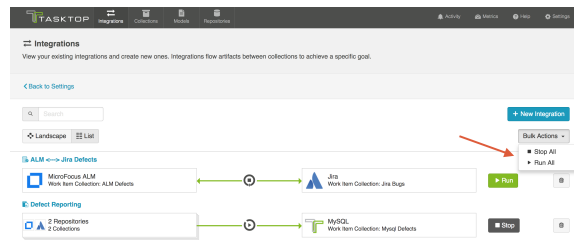
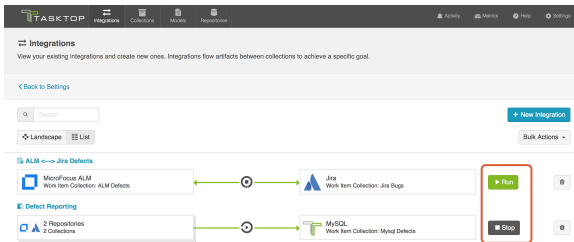
### From the Integration Configuration Screen

Simply click the **Run** to run the integration, and the **Stop** button to stop the integration.



## From the Integrations List Screen

Click **Run** or **Stop** next to each integration you would like to update. You can also use the **Bulk Actions** button to run or stop all integrations.

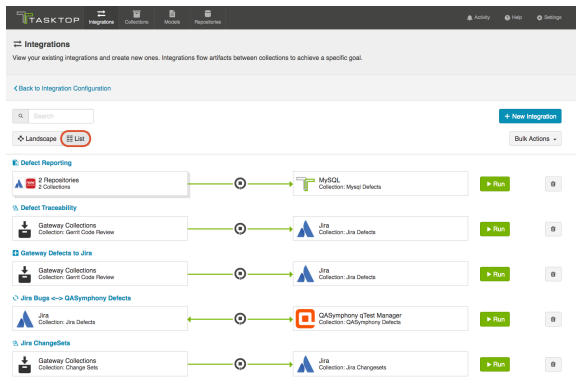
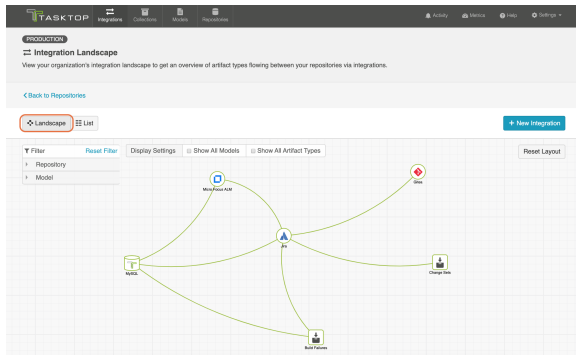


## Viewing Your Integrations

See [Tasktop Editions table](#) to determine if your edition contains Integration Landscape View functionality.



When viewing your integrations, you have the option of viewing them in either Landscape or List mode.



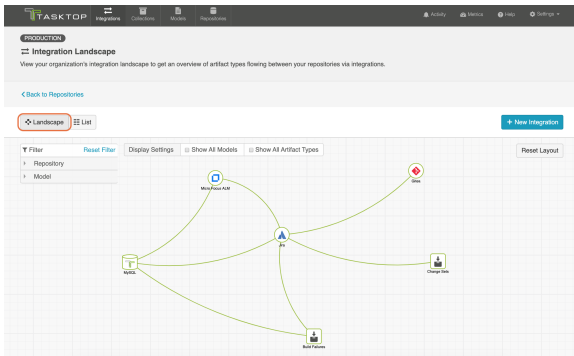
## Landscape View

See [Tasktop Editions table](#) to determine if your edition contains Integration Landscape View functionality.

Learn more about the Integration Landscape view in the video below:

Tasktop will default to the Landscape view, which enables you to visualize your entire integration landscape and see how your integrations relate to one another. Use our built-in filters to see as little or as much information as you'd like!

Here's a simplified view:

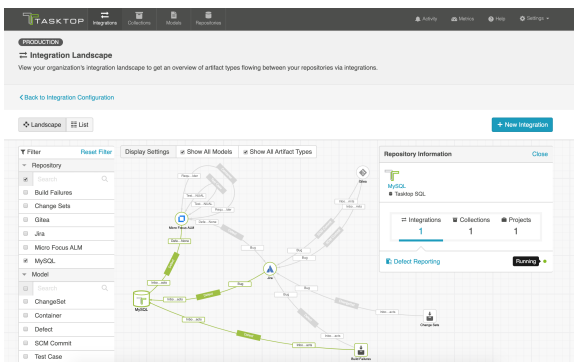


If you'd like to see additional information, you can utilize the filters, or click on a repository node to modify which information is shown.

Some examples of additional information you can see are:

- Models
- Artifact Types
- Artifact Creation Directionality Arrows
- List of all relevant integrations (see this by clicking on the repository node)
  - Indicator of whether each integration is running or not

Here's an example of a more detailed view:

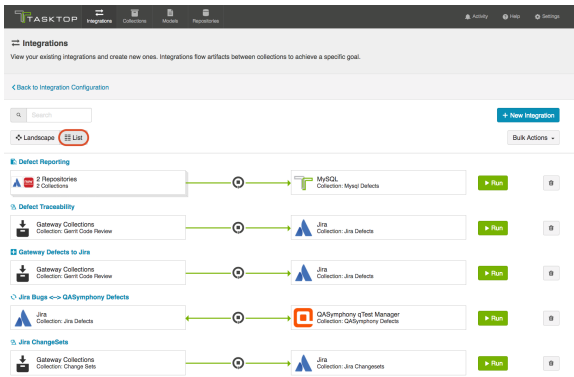


## List View

If you'd like, you can toggle to List view, which will show you a list of all integrations you have created.

You can use this view to:

- Start an Integration
- Stop an Integration
- Delete an Integration
- Click into an Integration and modify its configuration



## Reporting

### To ETL or Not To ETL?

ETL (Extract, Transform, Load) is a process where data is extracted from a database, transformed to be more suitable for reporting or analytics, and loaded into a database which is normally used for reporting.

The data structures populated directly by Tasktop are intended to be used as a source for ETL; Some kinds of reports are not easily produced without first performing an ETL process. ETL can also be beneficial for performance of reports.

Some reports are possible without first performing an ETL process. Examples of such reports include Artifact Cycle Time and Defect Count By State By Cycle Time.

### Example Reports

The following are examples of some reports that can be driven directly from the database tables populated by an Enterprise Data Stream Integration:

#### Artifact Cycle Time

Artifact Cycle Time is often a valuable metric to measure as it can help identify areas where efficiencies can be gained and ensure “lean flow”. We have provided a model called “Artifact Cycle Time” and can be used to easily flow the necessary data to your database – enabling you to create a variety of metrics and visualizations based on the cycle time of any artifact type.

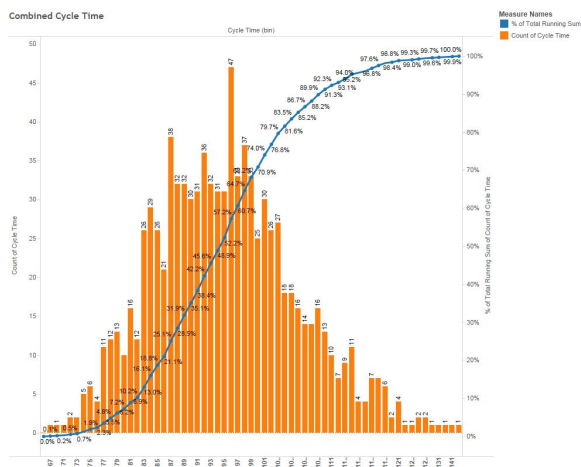
#### Artifact Cycle Time Model

Artifact Cycle Time
Formatted ID
Project

Type
Created
Modified
Severity
Status
Priority
Release
Assignee

If you use this model, you can easily produce visualizations such as a histogram that can identify the historical trend of cycle times.

### Artifact Cycle Time Histogram



### SQL

```

SELECT A.FORMATTED_ID, B.MODIFIED AS StatusOpen, C.MODIFIED AS StatusInProgress, D.MODIFIED AS
StatusReadyForTesting, E.MODIFIED AS StatusReadyForVerification, F.MODIFIED AS StatusComplete, G.MODIFIED AS
StatusShipped, A.STATUS AS CurrentStatus FROM ARTIFACT A
LEFT OUTER JOIN ARTIFACT B
ON B.ARTIFACT_ID = A.ARTIFACT_ID
AND B.STATUS = 'Open'
AND NOT EXISTS (SELECT * FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND (MODIFIED < B.MODIFIED OR
(MODIFIED = B.MODIFIED AND ID < B.ID)) AND STATUS = B.STATUS)
LEFT OUTER JOIN ARTIFACT C
ON C.ARTIFACT_ID = A.ARTIFACT_ID
AND C.STATUS = 'In Progress'
AND NOT EXISTS (SELECT * FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND (MODIFIED < C.MODIFIED OR
(MODIFIED = C.MODIFIED AND ID < C.ID)) AND STATUS = C.STATUS)
LEFT OUTER JOIN ARTIFACT D
ON D.ARTIFACT_ID = A.ARTIFACT_ID
AND D.STATUS = 'Ready for Testing'
AND D.MODIFIED > (SELECT MAX(MODIFIED) FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND STATUS IN
('Open', 'In Progress'))
AND NOT EXISTS (SELECT * FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND (MODIFIED < D.MODIFIED OR
(MODIFIED = D.MODIFIED AND ID < D.ID)) AND STATUS = D.STATUS
AND MODIFIED > (SELECT MAX(MODIFIED) FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND STATUS IN

```

```

('Open', 'In Progress'))
LEFT OUTER JOIN ARTIFACT E
  ON E.ARTIFACT_ID = A.ARTIFACT_ID
  AND E.STATUS = 'Ready for Verification'
  AND E.MODIFIED > (SELECT MAX(MODIFIED) FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND STATUS IN
('Open', 'In Progress', 'Ready for Testing'))
  AND NOT EXISTS (SELECT * FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND (MODIFIED < E.MODIFIED OR
(MODIFIED = E.MODIFIED AND ID < E.ID)) AND STATUS = E.STATUS
  AND MODIFIED > (SELECT MAX(MODIFIED) FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND STATUS IN
('Open', 'In Progress', 'Ready for Testing')))
LEFT OUTER JOIN ARTIFACT F
  ON F.ARTIFACT_ID = A.ARTIFACT_ID
  AND F.STATUS = 'Complete'
  AND F.MODIFIED > (SELECT MAX(MODIFIED) FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND STATUS IN
('Open', 'Ready for Testing', 'Ready for Verification'))
  AND NOT EXISTS (SELECT * FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND (MODIFIED < F.MODIFIED OR
(MODIFIED = F.MODIFIED AND ID < F.ID)) AND STATUS = F.STATUS
  AND MODIFIED > (SELECT MAX(MODIFIED) FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND STATUS IN
('Open', 'Ready for Testing', 'Ready for Verification')))
LEFT OUTER JOIN ARTIFACT G
  ON G.ARTIFACT_ID = A.ARTIFACT_ID
  AND G.STATUS = 'Shipped'
  AND G.MODIFIED > (SELECT MAX(MODIFIED) FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND STATUS IN
('Open', 'Ready for Testing', 'Ready for Verification', 'Complete'))
  AND NOT EXISTS (SELECT * FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND (MODIFIED < G.MODIFIED OR
(MODIFIED = G.MODIFIED AND ID < G.ID)) AND STATUS = G.STATUS
  AND MODIFIED > (SELECT MAX(MODIFIED) FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND STATUS IN
('Open', 'Ready for Testing', 'Ready for Verification', 'Complete')))
WHERE NOT EXISTS (SELECT * FROM ARTIFACT WHERE ARTIFACT_ID = A.ARTIFACT_ID AND (MODIFIED > A.MODIFIED OR
(MODIFIED = A.MODIFIED AND ID > A.ID)))
  AND (A.ARTIFACT_EVENT_TYPE IS NULL OR NOT A.ARTIFACT_EVENT_TYPE = 'removed')
ORDER BY A.FORMATTED_ID

```

The example above is designed to handle cases where an artifact is moved into a state more than once. For example, a defect that is moved to “Complete”, subsequently moved back into “In Progress”, then moved to “Complete” again is represented with a row having the second timestamp for the “Complete” status.

Artifact ID	Priority	Type	Severity	Repository	SubStatus	Status	Created	Modified	Event Type	Event Time	Event Description			
TC02-TestProject1	Low	Defect	3	HP	ALM	Open	2015-01-01 09:40:00	2015-01-04 10:10:00		2015-01-04 10:10:00	2015-04-27 08:18:00	2015-05-14 14:00:00	2015-06-01 06:37:00	Shipped
TC02-TestProject1	Medium	Defect	2	HP	ALM	Open	2015-02-01 14:00:00	2015-02-10 17:00:00		2015-02-10 17:00:00	2015-02-20 17:00:00	2015-04-01 17:00:00	2015-04-01 17:00:00	Shipped
TC02-TestProject1	High	Defect	1	HP	ALM	Open	2015-02-09 09:40:00	2015-02-13 10:00:00		2015-02-28 10:28:00	2015-03-09 08:40:00	2015-03-18 11:00:00	2015-03-27 10:00:00	Shipped
TC02-TestProject1	Low	Defect	3	HP	ALM	Open	2015-02-04 14:00:00	2015-02-10 16:00:00		2015-02-17 12:00:00	2015-02-17 14:00:00	2015-03-01 18:00:00	2015-04-01 14:00:00	Shipped
TC02-TestProject1	Medium	Defect	2	HP	ALM	Open	2015-02-07 19:40:00	2015-02-11 21:30:00		2015-02-15 10:41:00	2015-04-15 14:00:00	2015-04-28 10:00:00	2015-05-17 10:00:00	Shipped
TC02-TestProject1	High	Defect	1	HP	ALM	Open	2015-02-07 18:00:00	2015-02-10 10:00:00		2015-02-18 10:22:00	2015-02-20 10:00:00	2015-02-27 11:00:00	2015-03-17 18:00:00	Shipped
TC02-TestProject1	High	Defect	1	HP	ALM	Open	2015-02-12 19:00:00	2015-02-16 19:20:00		2015-04-02 10:00:00	2015-04-07 18:20:00	2015-04-14 14:00:00	2015-04-20 19:40:00	Shipped
TC02-TestProject1	High	Defect	1	HP	ALM	Open	2015-02-15 19:00:00	2015-02-19 10:00:00		2015-04-14 00:00:00	2015-04-21 10:00:00	2015-04-24 17:00:00	2015-04-24 00:00:00	Shipped

Reports can be driven from the results of this SQL query, subtracting dates to produce cycle times for the desired transitions (e.g. “Open” to “Shipped”).

Status values in the SQL above correspond to the values present in the “Artifact” model; repository-specific status values can be mapped to the model values in the corresponding collection field mapping. If status values are added, removed, or changed in the artifact model, then the SQL will have to be modified accordingly.

## Defect Count By State By Cycle Time

Defect Count By State By Cycle Time provides a count of defects by cycle time for each status of an artifact.

In this example, the cycle time is measured in days. Cycle time is only measured for status state transitions; Cycle time is not measured for the end state of an artifact.

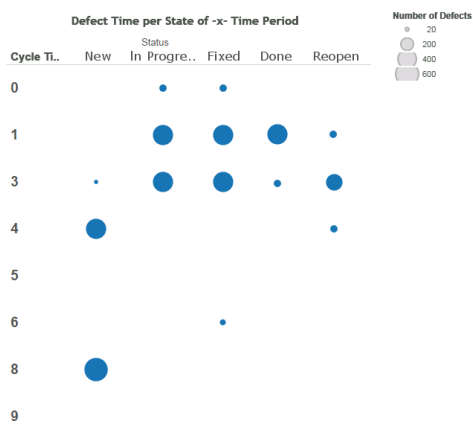
We provide a basic defect model packaged with our product:

## Basic Defect Model

Defect Model
Formatted ID
Project
Type
Created
Modified
Severity
Status
Summary
Summary-to-Description
Related Defects
Description

If you use this model, you can easily produce visualizations such as a bubble chart that can identify the volume of defects in each cycle time measured in days. This is simply a slightly different view into your overall cycle time.

## Cycle Time Volume



## SQL

```
SELECT status, COUNT(artifact_id), cycleTime FROM (
  SELECT A.ARTIFACT_ID AS artifact_id, A.STATUS AS status, SUM(
    TIMESTAMPDIFF(SQL_TSI_DAY,A.MODIFIED,B.
```

```

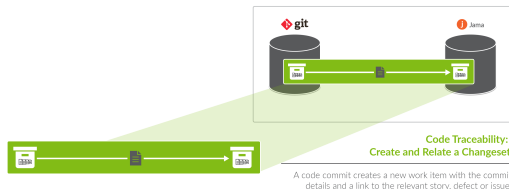
MODIFIED) ) AS cycleTime FROM DEFECT A
INNER JOIN DEFECT B ON A.ARTIFACT_ID = B.ARTIFACT_ID
AND A.ID != B.ID
AND A.STATUS != B.STATUS
AND A.MODIFIED <= B.MODIFIED
AND ((A.ARTIFACT_EVENT_TYPE IS NULL OR B.ARTIFACT_EVENT_TYPE IS NULL)
OR NOT (A.ARTIFACT_EVENT_TYPE = 'removed' OR B.ARTIFACT_EVENT_TYPE = 'removed'))
)
WHERE NOT EXISTS (
SELECT * FROM DEFECT C WHERE C.ARTIFACT_ID = A.ARTIFACT_ID AND C.ID != A.ID AND C.ID != B.ID
AND C.MODIFIED >= A.MODIFIED AND C.MODIFIED <= B.MODIFIED
AND ((C.STATUS = A.STATUS OR C.STATUS = B.STATUS) OR (C.STATUS != A.STATUS AND C.STATUS != B.STATUS))
)
AND NOT EXISTS (
SELECT * FROM DEFECT D WHERE D.ARTIFACT_ID = A.ARTIFACT_ID AND B.MODIFIED <= (
SELECT MAX(MODIFIED) FROM DEFECT D WHERE D.ARTIFACT_ID = A.ARTIFACT_ID AND D.ARTIFACT_EVENT_TYPE =
'removed'
)
)
)
GROUP BY A.ARTIFACT_ID, A.STATUS
) CT GROUP BY CT.status, CT.cycleTime
ORDER BY CT.status, CT.cycleTime

```

# Code Traceability: Create and Relate a Changeset

## What is a Code Traceability: Create and Relate a Changeset Integration?

*This integration template is only available in editions that have access to the Git repository.*



An *integration* is quite simply **the flow of information between two or more collections**.

A *Code Traceability: Create and Relate a Changeset* integration, specifically, creates new work items such as changesets or code commits in a repository such as Jama, when they are sent to Tasktop via an outbound only collection connecting to a repository such as Git.

These types of events are “fire and forget” - they create something new in your repository, but they don’t expect anything back. As such, they don’t mandate a full-blown two way synchronization; a lighter one-way integration can do the trick.

Here is an example of what you can do with the Code Traceability: Create and Relate a Changeset integration:

- Flow a Git code commit to a repository such as Jama as a changeset, and relate that changeset to an existing requirement or defect

When you configure your Code Traceability: Create and Relate a Changeset integration, you can customize the field flow, artifact filtering, and change detection for your integration.

## Use Case and Business Value

The Code Traceability: Create and Relate a Changeset template allows developers working in Source Control Management tools, such as Git, to flow artifacts, such as code commits, to a Requirements Management tool, such as Jama.

As part of the integration,



- Changesets, commit messages, or code reviews from an SCM tool, such as Git, will create corresponding changesets in Requirements Management tools, such as Jama
- Optionally, those newly created changesets can be related to their associated features, defects, or other artifacts in the Requirements Management tool

## Template Affordances

The Code Traceability: Create and Relate a Changeset Template allows you to flow artifacts in one direction: from your outbound only collection (i.e. an SCM tool, such as Git) to your work item collection (i.e. your Requirements Management tool).



## Before You Begin

Before you begin configuring your integration, you must configure your repository, model, and collections. Please review instructions below for each step

## Repository Configuration

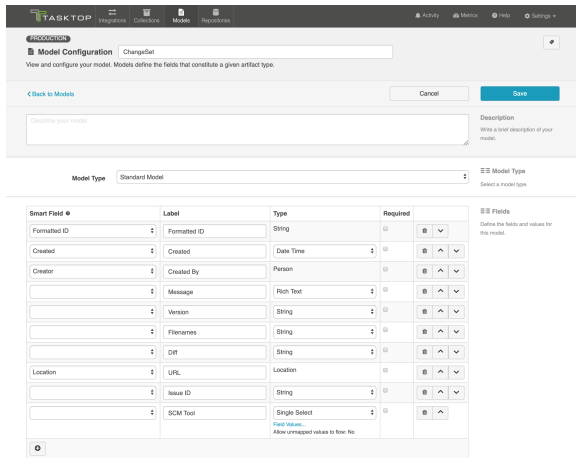
Please review the following pages to learn how to configure your repository:

- [Standard Repository Connection](#)
- Please review the Git Connector page in our [Connector Docs](#) for additional details on configuring the Git repository. This will serve as the source repository in your integration.

## Model Configuration

You can learn more about configuring your model here: [Step 2: Create or Reuse a Model](#)

Below is our recommended ChangeSet model configuration:



## Collection Configuration

To configure your source and target collections, please review the instructions below.

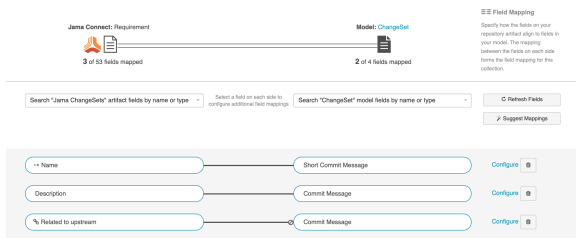
- To configure your source collection (i.e. your Git collection): [Outbound Only Collection](#)
- To configure your target collection (i.e. your Jama or Jira collection): [Work Item Collection \(Repository\)](#)
  - Please also review additional notes below

For your **target collection**,

- Ensure you are using the same model as your source collection (i.e. the ChangeSet model)

Configure the following mappings:

Source Repository (Git)	Model	Target Repository (i.e. Jama or Jira)
Short Commit Message (the first line of the commit message)	Short Commit Message (String or Rich text)	Summary /Name
Commit Message (the entire commit message)	Commit Message (String or Rich text) <i>If also mapping Commit Message to description as shown below, ensure transform is set to 'none' for the model on the target collection field mapping</i>	Relationship of choice (i.e. 'relates to')  see details below
Commit Message (the entire commit message)	Commit Message (String or Rich text)	Description



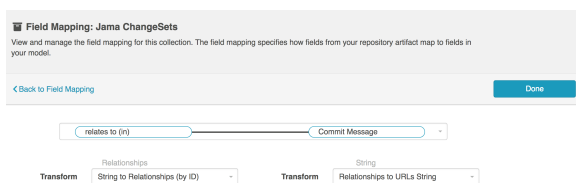
## Configuring Relationships for the Target Collection

**⚠** In order for your integration to run, there must be a mapping in your target collection that tells Tasktop how to handle relationships between artifacts. This must be done via a relationship-to-string (or relationship-to-rich text) field mapping in the target collection. If no such mapping exists, you will notice an issue on the [Activity Screen](#) that will block the integration from running.

To configure relationships for your target collection, go to the Field Mapping screen for that collection, and map the relationship type of choice (i.e., 'relates to') to the Commit Message string field in your model. The transform selected for this field mapping should be *String to Relationships (by ID)*. Tasktop will default to this transform.

Tasktop has built-in smarts to find any artifact IDs present in the commit message, and to then relate the newly created changeset in that target repository to the artifacts identified in that message. For example, if my Git code commit has 'ARTIFACT #123' listed in its commit message, if my relationship is mapped to the 'Commit Message' field in my model, when the corresponding changeset is created in my target repository, it will automatically include a relationship to the existing ARTIFACT #123 in that repository. This will use the *String to Relationships (by ID)* transformation.

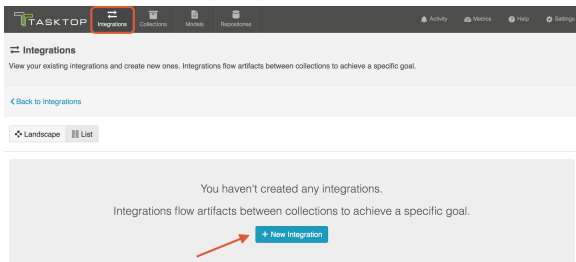
If a *relationship* field is mapped to the commit message, Tasktop will relate the newly created changeset to the first artifact ID it finds in the commit message (if there are multiple IDs). If a *relationships* field is mapped to the commit message, Tasktop will relate the newly created changeset to *all* artifact IDs it finds in the commit message.



## Configuring a Code Traceability: Create and Relate a Changeset Integration

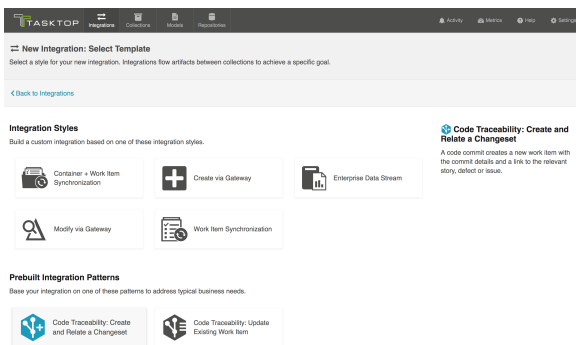
Now that you have all of your base components (i.e., repositories, models, and collections) set up, you can configure an integration to synchronize the artifacts in your collections.

To configure your integration, select **Integrations** at the top of the screen, then click **+ New Integration**.

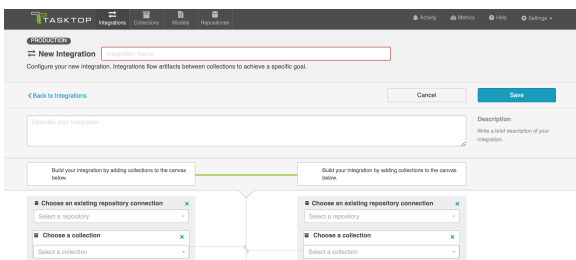


Select the Code Traceability: Create and Relate a Changeset template.

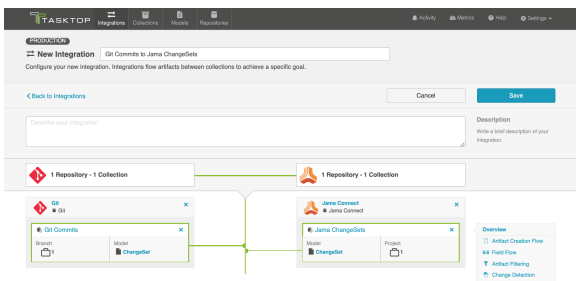
💡 Depending on the **edition** of Tasktop you are utilizing, you may not have all options available.



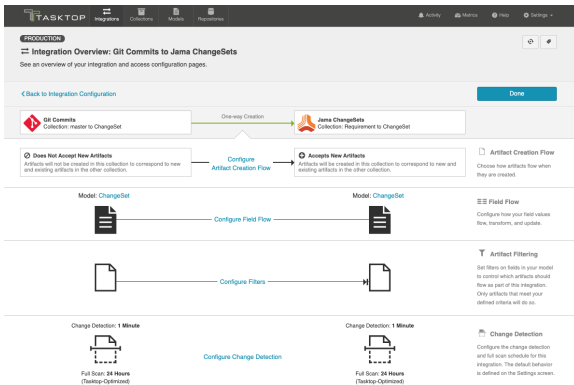
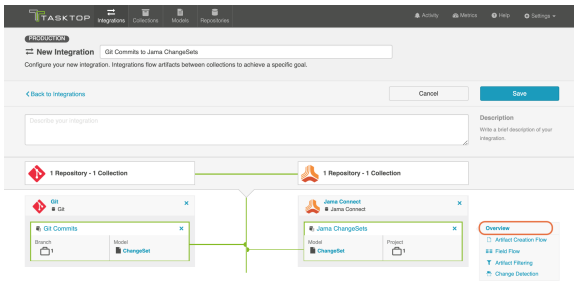
This will bring you to the New Integration screen.



Name your integration and select your repositories and collections.

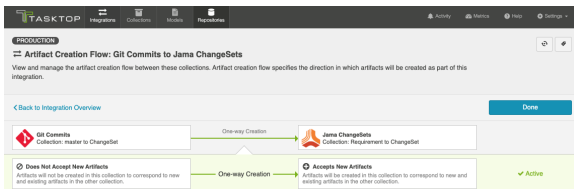


You can click the **Overview** link on the right side of the New Integration screen to get to the main display screen (shown in the second screenshot).



## Artifact Creation Flow

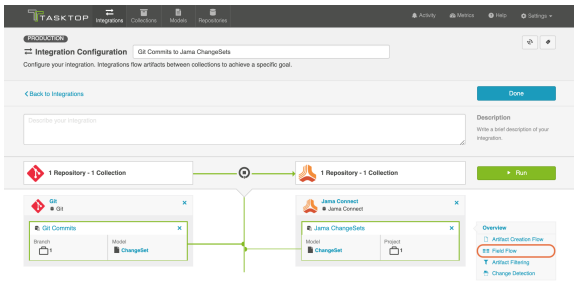
Artifacts will flow in one direction only: from the Git repository to the Work item repository. This cannot be modified.



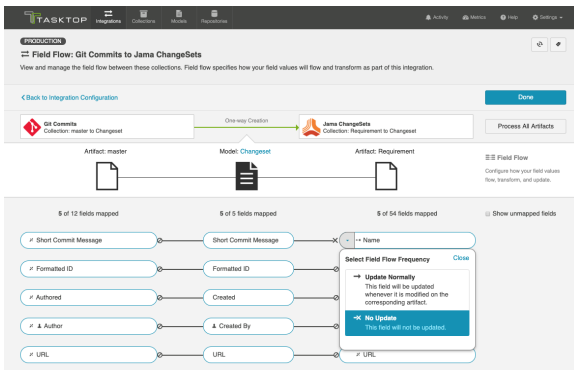
## Field Flow

The field flow configured for a given integration specifies which fields should flow in that integration. For *Code Traceability: Create and Relate a Changeset* integrations, you can choose to flow a given field (Update Normally) or to not flow a given field (No Update).

To get to the Field Flow screen, click **Field Flow** on the right pane of the Integration Configuration screen.



You will be directed to the Field Flow screen.










You can choose to flow a field ('update normally') or not flow it ('no update'). You'll notice that field flow goes in one direction only — from the outbound only collection *into* the repository collection.

You can see the names of the mapped artifact fields for each collection on the far left and far right, with the model fields displayed in the middle. By default, model fields without mapped repository fields are hidden. You can see all model fields by checking the 'Show unmapped fields' checkbox. Constant values will be identified by a grey box and the constant value icon.

## Field Flow Icons

On the Field Flow screen, you will see a number of icons, which will help you understand any special properties or requirements for each field. If you hover your mouse over an icon, you will see a pop-up explaining what the icon means. You can also review their meanings in the legend below:

Icon	Meaning
	<p>A constant value will be sent.</p> <p>Note that:</p> <ul style="list-style-type: none"> <li>If the icon is on the side of the collection, this means that a constant value will be sent to your model. This means that any time this collection is integrated with another collection, the <i>other</i> collection will receive this constant value for the field in question.</li> <li>If the icon is on the side of the model, this means a constant value will be sent to your collection. This means that any time this collection is integrated with another collection, that <i>this</i> collection will receive this constant value for the field in question.</li> </ul>

	<p>A state transition will be utilized. Note that:</p> <ul style="list-style-type: none"> <li>• If the icon is on the side of the collection, this means that a <a href="#">state transition graph</a> is being utilized.</li> <li>• If the icon is on the side of the model, this means that a <a href="#">state transition extension</a> is being utilized.</li> </ul> <p> Note that Tasktop will update the field flow frequency for fields utilizing state transitions to 'no update.' This is because they are updated via the transition and not via normal 'field flow.' Do not modify the field flow frequency for this field.</p>
	<p>Collection field is read-only and cannot receive data</p>
<p>←*</p> <p>*→</p>	<p>To create artifacts in your collection, this field must be mapped to your model.</p>
	<p>This is a required field in your model; it must be mapped to your collection.</p>
	<p>This field will not be updated as part of your integration, due to how you have configured it. This field flow configuration can be changed if you'd like.</p>
	<p>This field will not be updated as part of your integration because the mapping would be invalid. You do not have the option of changing this.</p>
	<p>This field will update normally as part of your synchronize integration; this means it will be updated whenever it is modified on the corresponding artifact.</p>

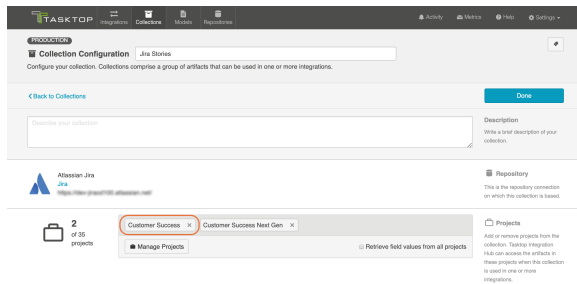
## Artifact Routing

Artifact routing is applied based on the artifact IDs included in each commit message. It cannot be configured or modified in Tasktop.

- New code commits or changesets will route to the project containing the first related artifact. For example, if your code commit references ARTIFACT #123, which resides in Project A in Jama, your newly created Jama changeset will be created in Project A as well. If Project A is not a part of your collection, the artifact will either be created in the intended project (if the external repository allows this), or it will be created in the first project in the collection\*
- If your code commit or changeset does not reference an artifact in your target repository, it will be created in the first project listed in the collection\*

\*Here, by 'first' we mean the first project listed on the Collection Configuration screen. Tasktop will list all projects added at the same time alphabetically. Then once saved, it will add any new projects added under subsequent saves after the initial list of projects.

**Note:** For routing to work in a Code Traceability: Create and Relate a Changeset integration with Jama, the commits and their related artifacts must exist in Jama Sets adjacent to each other and must always retain a shared exact parent. Otherwise you will get an error. If the commit does not have a related artifact, they will be created in the root Set of the project in Jama.



## Artifact Filtering

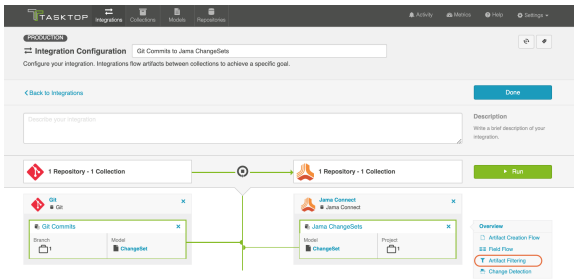
**Artifact Filtering** enables you to set filters in order to limit which artifacts are eligible to flow in an integration.

To use a field for artifact filtering, it must:

- Be a part of your model, and be mapped to the collection you are filtering from
- Be one of the following field types:
  - Single Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **only** fields that are already a part of the model will be considered for artifact filtering
  - Multi-Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **only** fields that are already a part of the model will be considered for artifact filtering
  - Date
  - Date/Time
  - Duration
  - String

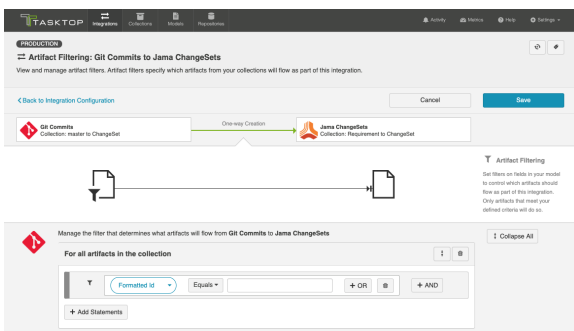
To configure Artifact Filtering, select **Artifact Filtering** from the right pane of the Integration Configuration screen:





This will lead you to the Artifact Filtering Configuration screen, where you can configure one or more criteria for artifact filtering.

💡 You can click the **Collapse All** button to view an easier-to-read summary of your artifact filtering statements.



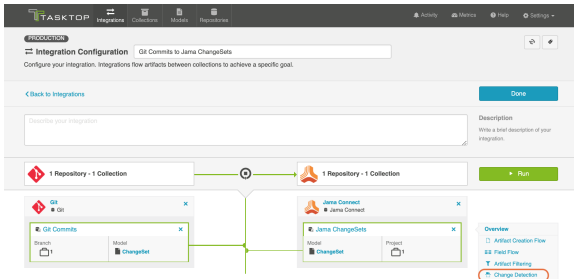
## Change Detection

Tasktop's default global change detection settings can be found on the [General \(Settings\)](#) screen. However, if you'd like to override the global defaults, you can configure integration-specific change detection and full scan intervals by clicking the **Change Detection** link.

The **Change Detection Interval** is the time between polling requests to detect *only changed artifacts*. This defaults to 1 minute on the General (Settings) screen, but can be customized as desired.

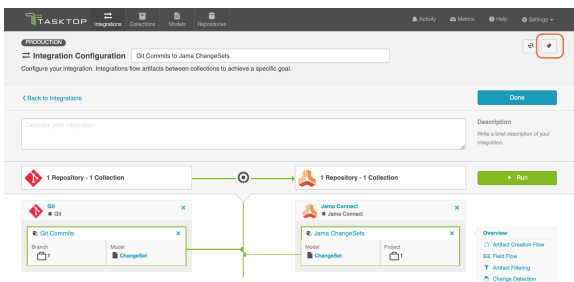
The **Full Scan Interval** is the time between polling requests to detect changed artifacts, in which *all* artifacts of a collection are scanned. Not all changes to an artifact will register as a change. Some repositories do not mark items as changed when (for example) a relationship is added or an attachment has changed. These may not be picked up by regular Change Detection, but will be picked up by a Full Scan. This defaults to 24 hours on the General (Settings) screen, but can be customized as desired. Users can also customize the full scan style for each integration to impact performance and server load, based on their integration and repository configuration.

To configure integration-specific change detection, click the **Change Detection** link. You can find details on this process [here](#). Note that for a *Code Traceability: Create and Relate a Changeset*, change detection can only be updated for the source (Git) collection.



## Viewing Associated Configuration Elements

To view associated configuration elements (such as collections or models that utilize the integration you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



### Associated Elements for Integration "Git Commits to Jama ChangeSets"

- 2 Repository Collections used by this Integration**
  - [Git Commits](#)
  - [Jama ChangeSets](#)
- 1 Model used by this Integration**
  - [ChangeSet](#)
- 2 Repository Connections used by this Integration**
  - [Git](#)
  - [Jama Connect](#)

Close

## Running Your Integration

To run your integration, please see details here: [Running Your Integration\(s\)](#)

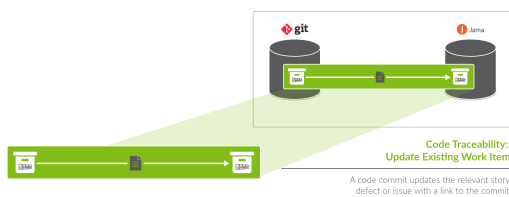
## Viewing Your Integration

To view your integration, please see details here: [Viewing Your Integration\(s\)](#)

# Code Traceability: Update Existing Work Item

## What is a Code Traceability: Update Existing Work Item Integration?

*This integration template is only available in editions that have access to the Git repository.*



An *integration* is quite simply **the flow of information between two or more collections.**

A *Code Traceability: Update Existing Work Item* integration, specifically, flows information from an outbound only collection (such as Git Commits) to a field on an existing artifact in a work item collection (such as Jama Codes).

These types of events are “fire and forget” - they create something new in your repository, but they don’t expect anything back. As such, they don’t mandate a full-blown two way synchronization; a lighter one-way integration can do the trick.

Here are some examples of what you can do with the Code Traceability: Update Existing Work Item integration:

- Flow Git code commit information to a custom field on a Jama code artifact
- Flow code commit information from Git hosting services such as Bitbucket, Gerrit, and more to a custom field on an associated requirement, defect, or epic in an Agile Planning or Requirements Management tool

When you configure your Code Traceability: Update Existing Work Item integration, you can customize the field flow, artifact filtering, and change detection configuration of your integration.

## Use Case and Business Value

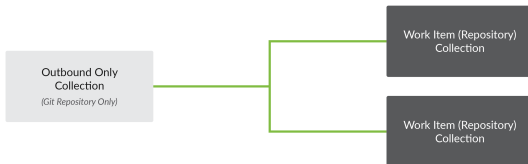
The Code Traceability: Update Existing Work Item template allows developers working in Source Control Management tools, such as Git, to flow data from code commits to an Agile Planning tool, such as Jira, to record information from the commit message directly on the related defect or feature.

As part of the integration,

- Changesets, commit messages, or code reviews from an SCM tool, such as Git, will flow information to a field on an artifact in a Requirements Management Planning tool, such as Jama

## Template Affordances

The Code Traceability: Update Existing Work Item template allows you to flow artifacts in one direction: from your outbound only collection (i.e. your Gti Commits collection) to your work item collection (i.e., your Requirements Management artifacts).



## Before You Begin

Before you begin configuring your integration, you must configure your repository, model, and collections. Please review instructions below for each step

## Repository Configuration

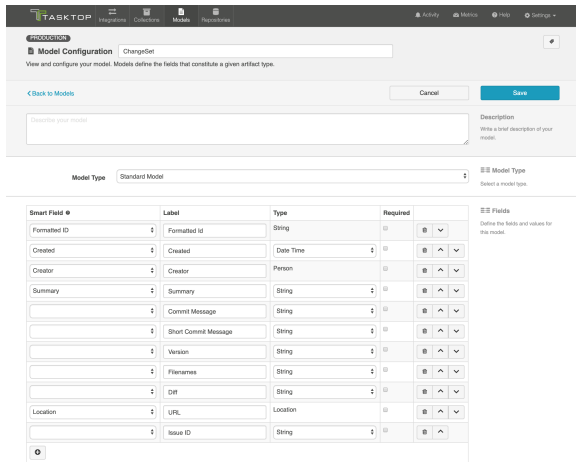
Please review the following pages to learn how to configure your repository:

- [Standard Repository Connection](#)
- Please review the Git Connector page in our [Connector Docs](#) for additional details on configuring the Git repository. This will serve as the source repository in your integration.

## Model Configuration

You can learn more about configuring your model here: [Step 2: Create or Reuse a Model](#)

Below is our recommended ChangeSet model configuration:



## Collection Configuration

To configure your source and target collections, please review the instructions below.

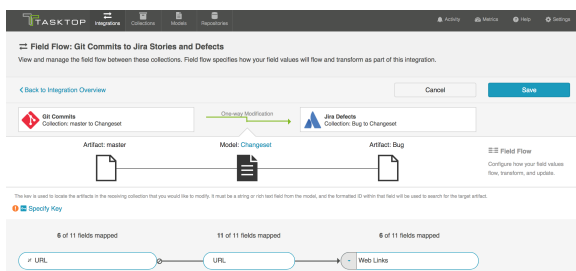
- To configure your source collection (i.e., your Git collection): [Outbound Only Collection](#)
- To configure your target collection (i.e., your Jama or Jira collection): [Work Item Collection \(Repository\)](#)
  - Please also review additional notes below

For your **target collection**,

- Ensure you are using the same model as your source collection (i.e., the Changeset model)

Configure the following mappings:

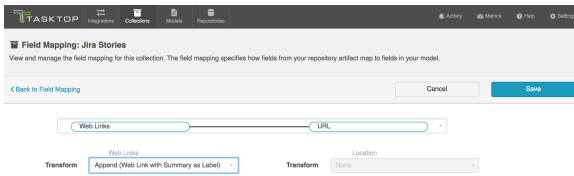
Source Repository (Git)	Model	Target Repository (i.e. Jama or Jira)
URL	URL	Web Link



## Configuring Commit Links

To flow a Git commit link to the artifact in the target repository, you must map the URL model field to the string, rich text, URL, or Web Link field in your target repository. If your target repository supports web links with labels, you'll see that you can configure a 'Location to Web Link (Summary as Label)' or 'Append (Web Link with Summary as Label)' transform. In most cases, you will want to select 'Append

(Web Link with Summary as Label),' as this will allow you to flow a link to each related commit, with each link using that commit's 'short commit message' (summary) as its label.



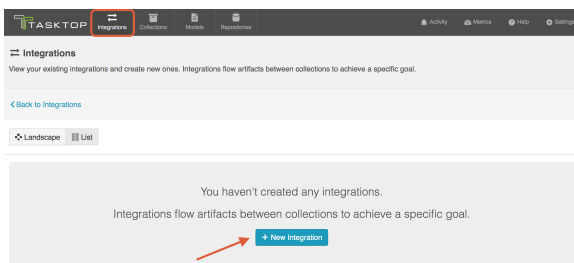
## Important Note about Field Updates

⚠ Since this integration type will update existing artifacts in your target repository, be aware that any field mappings configured in your Field Flow will update fields on that existing artifact. As such, you should ensure that only fields that you'd like to update are set to flow. For example, you likely will not want to overwrite the summary or description fields in your target collection. Most likely, the only fields of concern will be the field that you are flowing the commit link to (i.e., the URL or Web Link field).

## Configuring a Code Traceability: Update Existing Work Item Integration

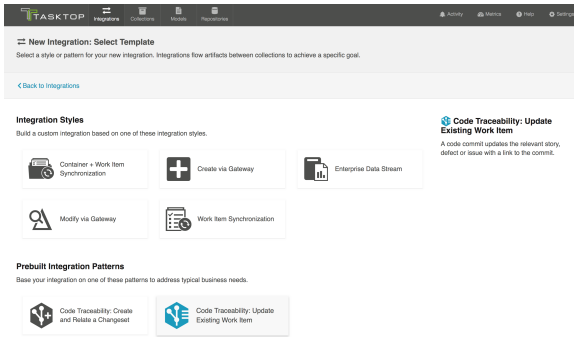
Now that you have all of your base components (i.e. repositories, models, and collections) set up, you can configure an integration to synchronize the artifacts in your collections.

To configure your integration, select **Integrations** at the top of the screen, then click **+ New Integration**.

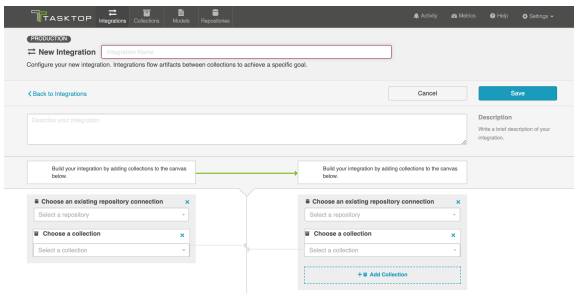


Select the **Code Traceability: Update Existing Work Item** template.

💡 Depending on the [edition](#) of Tasktop you are utilizing, you may not have all options available.



This will bring you to the **New Integration** screen.

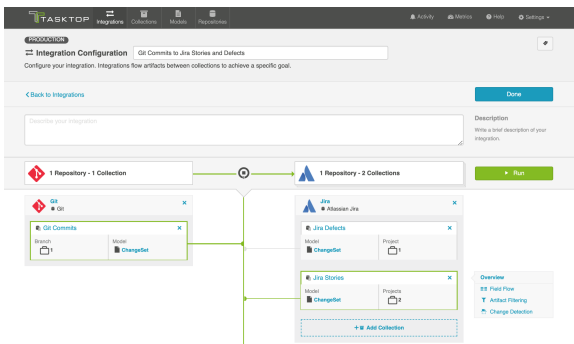


Name your integration and select your repositories and collections.

Note that you can add multiple target collections within the same repository if you'd like to flow commit information to multiple artifact types.

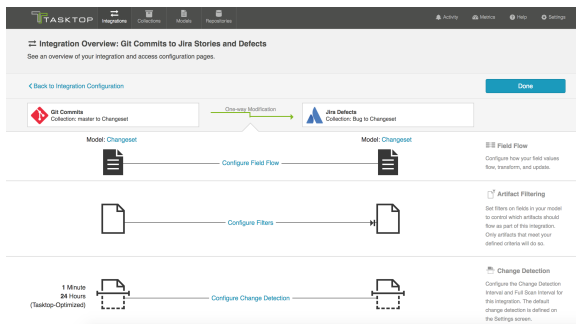
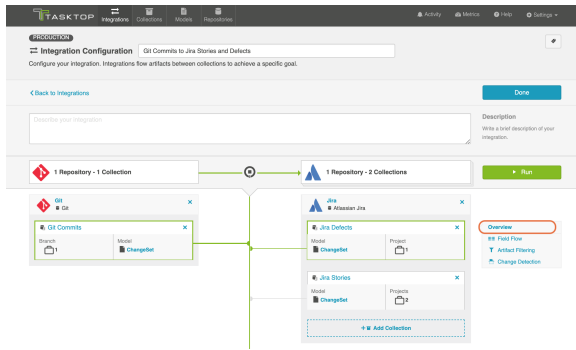
Click **Save**.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.



You can click the **Overview** link on the right side of the Integration Configuration screen to get to the main display screen (shown in the second screenshot).

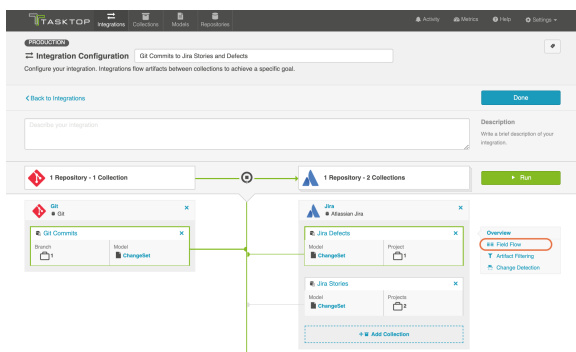
**Note:** The Overview screen will only show two collections at a time — one source collection and one target collection. If there are multiple target collections in your integration, make sure the one you are interested in is selected before clicking **Overview**.



## Field Flow

The field flow configured for a given integration specifies which fields should flow in that integration. For *Code Traceability: Update Existing Work Item* integrations, you can choose to flow a given field (Update Normally) or to not flow a given field (No Update).

To get to the Field Flow screen, select your desired collections and click **Field Flow** on the right side of the Integration Configuration screen.



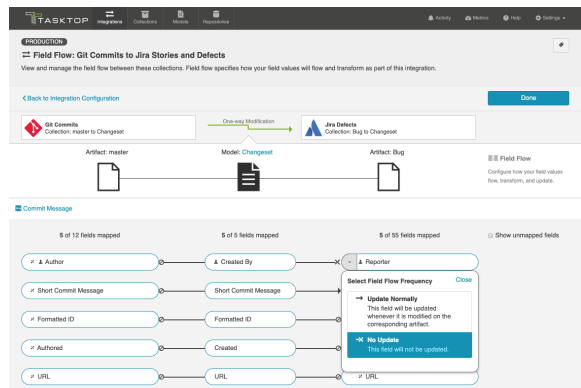
## Important Note about Field Updates

⚠ Since this integration type will update existing artifacts in your target repository, be aware that any field mappings configured in your Field Flow will update fields on that existing artifact. As such, you should ensure that only fields that you'd like to update are set to flow. For example, you likely will not want to overwrite the summary or description fields. Most likely, the only fields of concern will be the field that you are flowing the commit link to (i.e. the URL or Web Link field).



Note that in our example, only the URL field is set to flow into the target repository. Git will flow a web link for any related commits to a field on the Jira artifact, but will not overwrite any other Jira fields such as summary, description, etc.

If needed, you can manually set other fields not to flow.

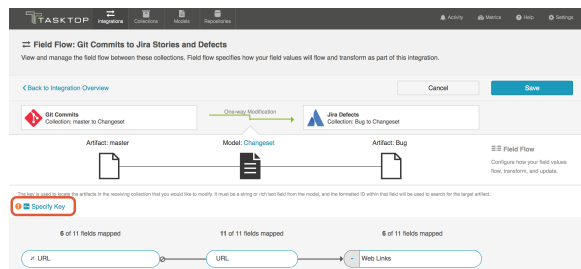


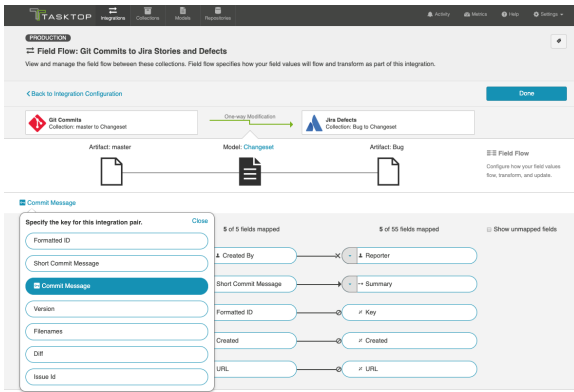
You can see the names of the mapped artifact fields for each collection on the far left and far right, with the model fields displayed in the middle. By default, model fields without mapped repository fields are hidden. You can see all model fields by checking the **Show unmapped fields** checkbox. Constant values will be identified by a grey box and the constant value icon.

## Specifying Your Key

In order for your integration to run successfully, you will need to specify your key if one is not specified yet. If possible, Tasktop will set the key as the Commit Message field in Git. If that field is not mapped, Tasktop will set the key as the Summary field. If neither field is mapped, you will need to select a field to use as the key.

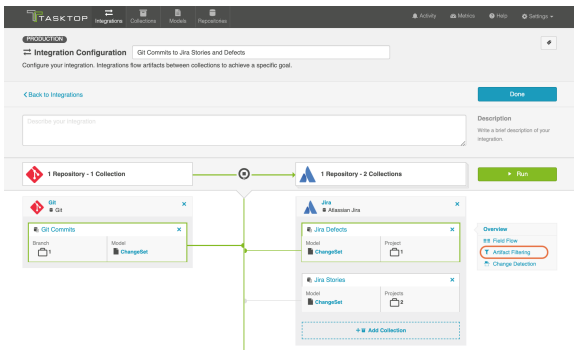
The key is used to locate the artifacts in the target collection that you would like to modify. It must be a string or rich text field from the model, and the formatted ID within that field will be used to search for the target artifact.





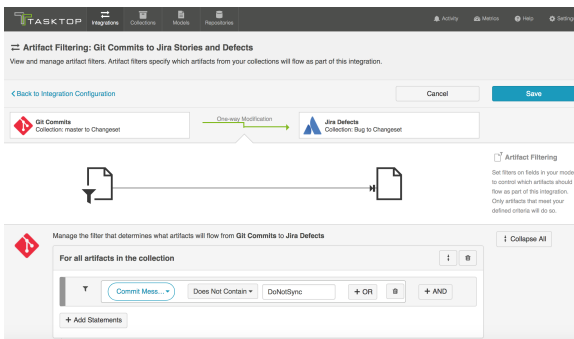
Page not found for multiexcerpt macro.

The page: **Code Traceability: Create and Relate a Changeset** was not found. Please check/update the page name used in the 'multiexcerpt-include' macro.



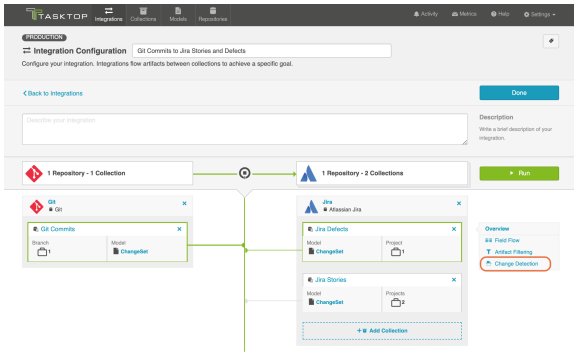
This will lead you to the Artifact Filtering Configuration screen, where you can configure one or more criteria for artifact filtering.

💡 You can click the **Collapse All** button to view an easier-to-read summary of your artifact filtering statements.



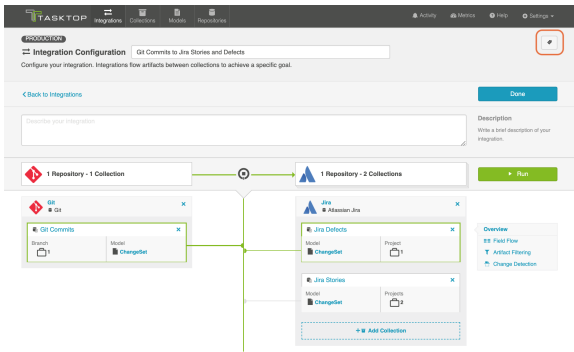
Page not found for multiexcerpt macro.

The page: **Code Traceability: Create and Relate a Changeset** was not found. Please check/update the page name used in the 'multiexcerpt-include' macro.



## Viewing Associated Configuration Elements

To view associated configuration elements (such as collections or models that utilize the integration you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



### Associated Elements for Integration "Git Commits to Jira Stories and Defects"

- 3 Repository Collections used by this Integration**
  - [Git Commits](#)
  - [Jira Defects](#)
  - [Jira Stories](#)
- 1 Model used by this Integration**
  - [Changeset](#)
- 2 Repository Connections used by this Integration**
  - [Git](#)
  - [JIRA](#)




Close

Page not found for multiexcerpt macro.

The page: **Code Traceability: Create and Relate a Changeset** was not found. Please check/update the page name used in the 'multiexcerpt-include' macro.

# Test Synchronization

See [Tasktop Editions table](#) to determine if your edition contains Test Synchronization functionality.

	Supported Repository	Supported Tasktop Version
	Micro Focus ALM/Quality Center	Tasktop Integration Hub: 19.2 and later
	Micro Focus ALM Octane	Tasktop Integration Hub: 20.4 and later
	Tricentis Tosca	Tasktop Integration Hub: 19.2 and later
	Jama Connect	Tasktop Integration Hub: 21.3 and later
	TestRail	Tasktop Integration Hub: 21.4 and later

## Introduction

Tasktop Integration Hub offers integration solutions to flow test artifacts such as test results, test steps, and their associated tests, test runs, test instances, and folder structures. Please review sections below to learn more about supported test scenarios in Tasktop.

## Test Step Synchronization

Test Step synchronization is currently supported for the following integration scenarios:

- ALM Test Steps ALM Test Steps
- ALM Test Steps Octane Test Steps
- Jama Test Steps Octane Test Steps
- Octane Test Steps Jama Test Steps
- Jama Test Steps Jama Test Steps
- TestRail Test Steps Jama Test Steps
- Jama Test Steps TestRail Test Steps

Test Step synchronization is currently supported for the following artifacts:

- Design Steps on ALM Tests
- Run Steps on ALM Test Runs

---

## Micro Focus ALM

Use the instructions below to configure the following integrations:

- ALM Test Steps ALM Test Steps

## Step 1: Connect to your Repository

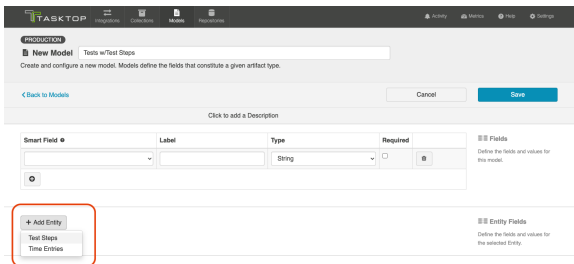
First, connect to your repository by following the instructions [here](#).

You can learn more about ALM-specific configuration in our [Connector Docs](#).

## Step 2: Construct your Model

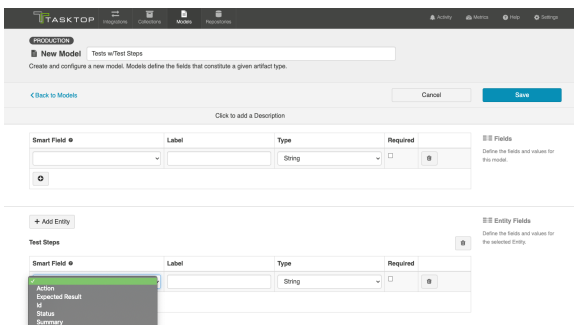
To flow test steps, you will need to add the **Test Steps** entity when creating your Test model.

To do this, click **+ Add Entity** and select the **Test Steps** option.

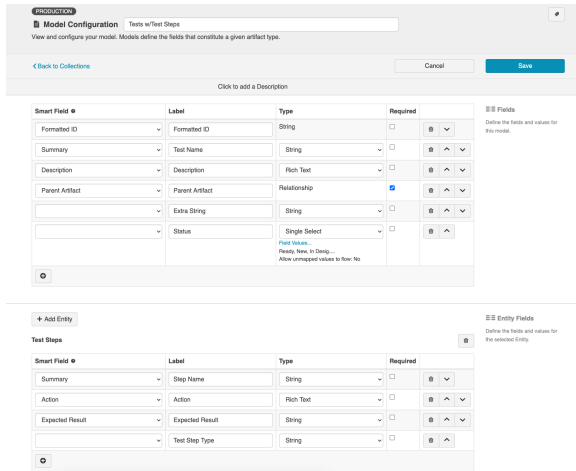


A Test Steps entity will be added. On the Model Configuration screen, you will then see two panels:

1. **Test Fields:** In the top section, add any fields you'd like to flow on the test (or test run) artifact that are not part of its associated test steps.
2. **Test Step Fields:** In the bottom section, add any fields you'd like to flow that are a part of test steps. You'll see that Tasktop provides some Smart Fields that are test step specific to help you get started, but you can add any other desired fields by leaving the Smart Field blank.



Here is an example of a very simple Test Model with Test Steps:



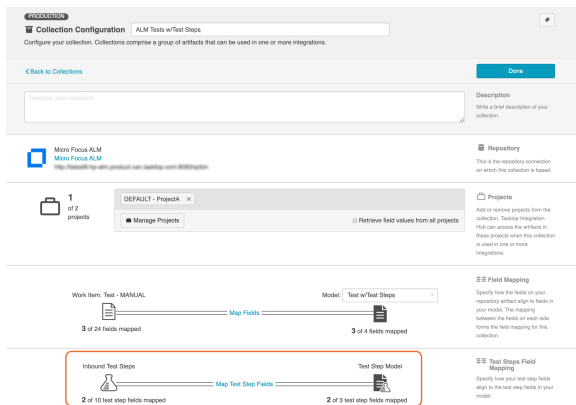
You can find general details on how to create a model [here](#).

## Step 3: Create your Collection

You can find general details on how to create a collection [here](#).

You will see a **Map Test Fields** sash on your collection if:

- Your model is a Model with Test Steps, and
- Your artifact is an ALM Test, ALM Test Run



The process to map test step fields is very similar to the process on the normal [Field Mapping](#) screen. Note that both relationship(s) and other field types for test steps will be mapped on this one sash.

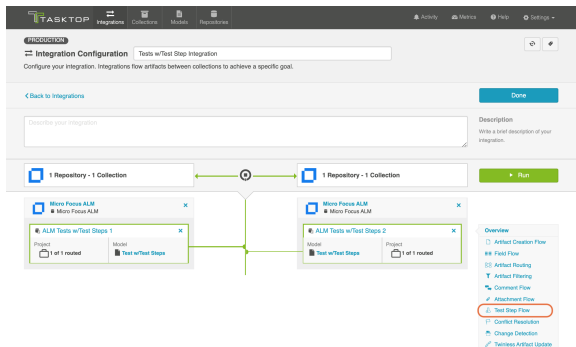
💡 Tests and Test Steps do not require a typical relationship field mapping to link them. We've added behind-the-scenes smarts to couple them for you.

## Step 4: Configure your Integration

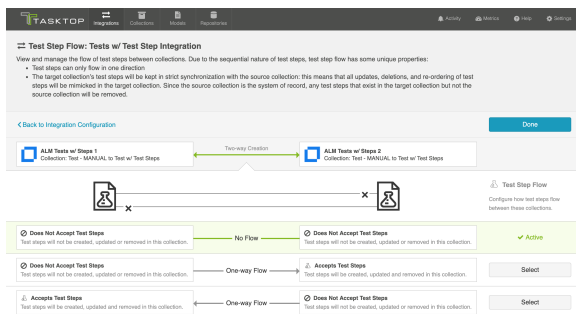
You can find general details on how to configure an integration [here](#).

In order to see a Test Step Flow link on the integration configuration screen, the following conditions must be met:

- Model is of type 'Model with Test Steps'
- Artifacts in both collections have test steps
- The relevant Tasktop connectors must support test steps (see [Connector Documentation](#) to confirm)



Clicking the link will bring you to the Test Step Flow screen, where you can click **Select** to choose your desired Test Step Flow style:



**i** Due to the sequential nature of test steps, test step flow has some unique properties:

- Test steps can only flow in one direction
- The target collection's test steps will be kept in strict synchronization with the source collection: this means that all updates, deletions, and re-ordering of test steps will be mimicked in the target collection. Since the source collection is the system of record, any test steps that exist in the target collection but not the source collection will be removed.
- If the test steps on the target artifact are changed by an end-user, they will be updated by Tasktop when one of the fields or ordering on the source artifact's test steps is changed.

**Note:** Comments and attachments are not currently supported on Test Steps.

## Micro Focus ALM Octane

Use the instructions below to configure the following integrations:

- ALM Test Steps Octane Test Steps

## Test Architecture

Before you begin configuring your integration, it's important to understand how test artifacts relate to one another.

While the goal of this integration is to flow test results, the architecture required to do so is more complex than one might assume.

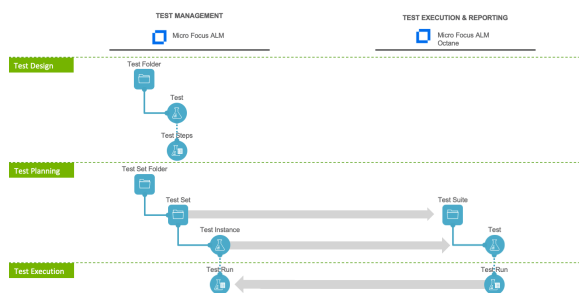
In this integration scenario, the challenge is synchronizing test steps between the two tools:

- In ALM, **Test Steps** are sub-entities of the **Test** artifact and referenced by a specific **Test Instance**.
- In ALM Octane, **Test Steps** are fields on the **Test** artifact.

The goal is to synchronize the ALM **Test Instance** artifact with the ALM Octane **Test** artifact, while including the **Test Steps** that exist on the related ALM **Test** artifact.

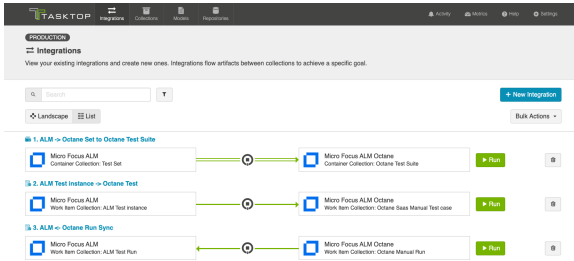
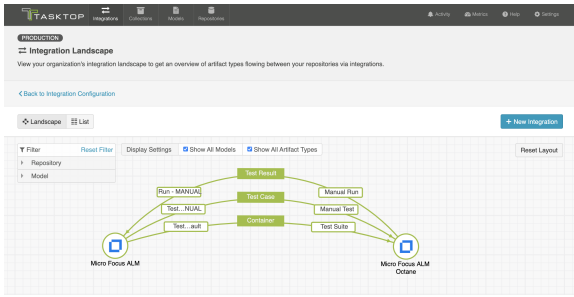
But don't worry — it only requires three integrations:

Integration	Container A	Container B	Work Item
Test Design	Test Set	Test Suite	--
Test Planning	--	--	Test Instance
Test Execution	--	--	Test Run



Once configured, your integrations will look like the images below.





**Tip:** To keep your integrations in order, we recommend appending a number to the beginning of each title (i.e., "1 - Test Design," "2 - Test Planning," "3 - Test Execution").

Now that you've familiarized yourself with the test architecture for this integration scenario, let's get started!

## Step 1: Connect to Your Repository

First, connect to your repository by following the instructions [here](#).

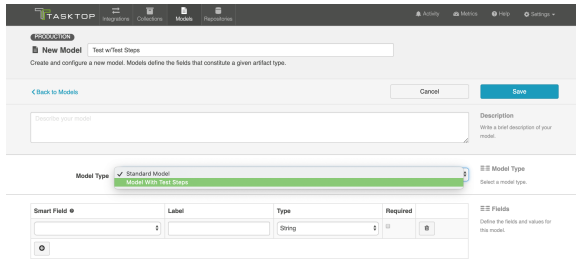
You can learn more about ALM-Octane specific configuration in our [Connector Docs](#).

## Step 2: Construct your Model

To flow test steps, you must add the **Test Steps** entity when creating your Test model.

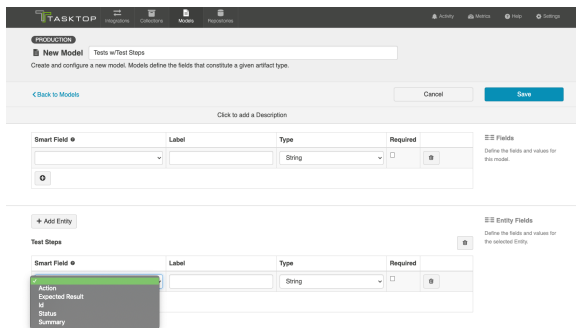
To do this, click **+ Add Entity** and select the **Test Steps** option.

Smart Field	Label	Type	Required
		String	<input type="checkbox"/>

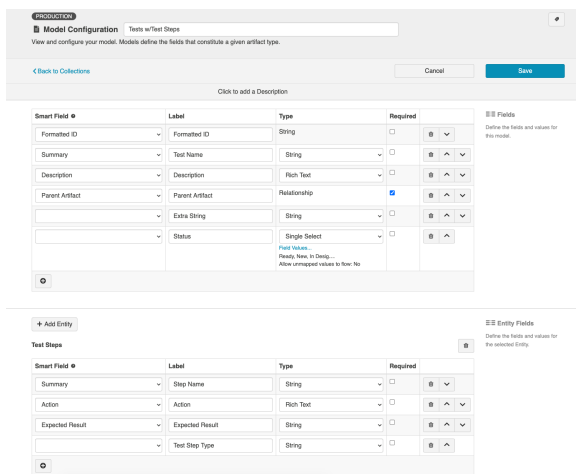


A **Test Steps** entity will be added to your model. On the Model Configuration screen, you will then see two panels:

1. **Test Fields:** In the top section, add any fields you'd like to flow on the test (or test run) artifact that are not part of its associated test steps.
2. **Test Step Fields:** In the bottom section, add any fields you'd like to flow that are a part of test steps. You'll see that Tasktop provides some Smart Fields that are test step specific to help you get started, but you can add any other desired fields by leaving the Smart Field blank.



Here is an example of a very simple Test Model with Test Steps:



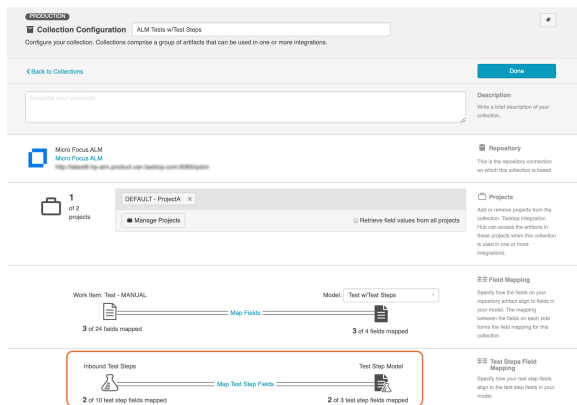
You can find general details on creating a model [here](#).

## Step 3: Create your Collection

You can find general details on how to create a collection [here](#).

You will see a **Map Test Fields** sash on your collection if:

- Your model is a Model with Test Steps, and
- Your artifact is an ALM Test or Octane Test



The process to map test step fields is very similar to the process on the normal [Field Mapping](#) screen. Note that both relationship(s) and other field types for test steps will be mapped on this one sash.

💡 Tests and Test Steps do not require a typical relationship field mapping to link them. We've added behind-the-scenes smarts to couple them for you.

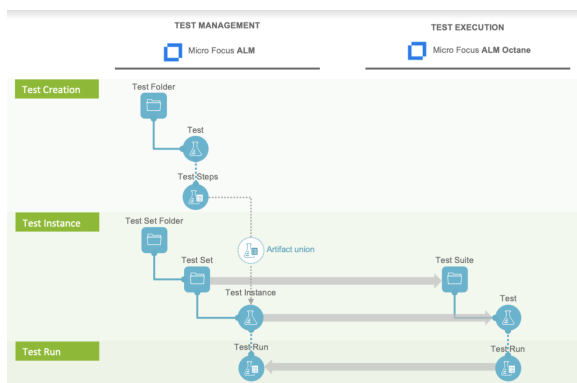
## Step 4: Configure an Artifact Union

To flow Test Steps from ALM to Octane, you will need to configure an artifact union between ALM Test Instances and ALM Tests so that the Test Steps will flow into the Octane Tests collection.

Artifact unions specify the related artifacts whose fields may flow along with the artifacts in your collection.

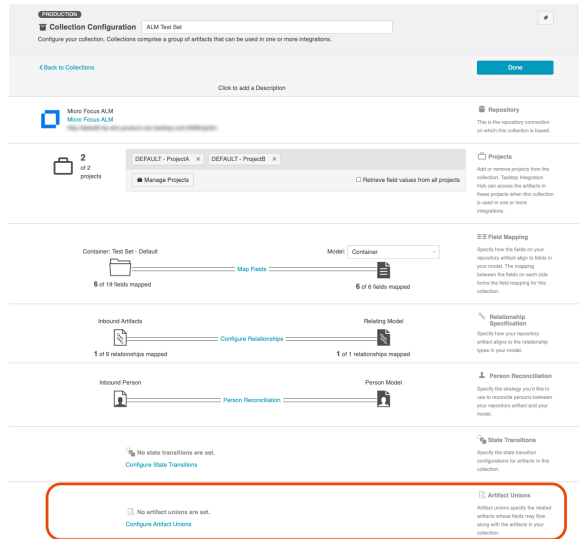
As ALM Test Steps only exist on the ALM 'Test' artifact, test steps cannot flow from ALM to Octane. With the help of an artifact union, through a shared relationship field between ALM Test Instances and ALM Tests, the test steps are able to flow.

Learn more about artifact unions [here](#).

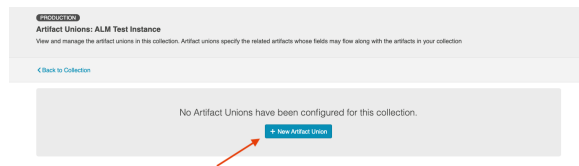


To access artifact unions, navigate to the Collection Configuration screen and click **Configure Artifact Unions**.

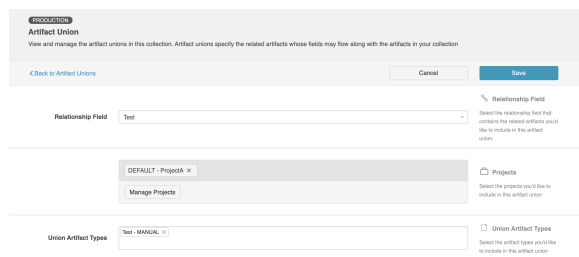
💡 The Artifact Unions sash will only be visible when there is a single relationship/container field available for the collection.



Navigate to the ALM Test Instance collection and create a new artifact union.



Then, select the **Test** relationship field, desired project(s), and artifact type.



After saving the artifact union, click **Map Test Step Fields** on the Collection Configuration screen.

On this screen, select the **Test** artifact union from the **Flow Test Steps From** dropdown menu.

After you've selected where you'd like the test steps to flow from, click **Save**.

**Tip:** Clicking **ctrl+s** on Windows and **cmd+s** on macOS will save your configuration.

The next step will be to configure the necessary integrations.

## Step 5: Configure your Integrations

You can find general details on how to configure an integration [here](#).

In order to see a Test Step Flow link on the integration configuration screen, the following conditions must be met:

- Model is of type 'Model with Test Steps'
- Artifacts in both collections have test steps
- The relevant Tasktop connectors must support test steps (see [Connector Documentation](#) to confirm)

### Integration 1: Test Design

The first integration you will configure is a [Container + Work Item Synchronization](#) flowing **Test Sets /Test Suites** (container).



#### Supported Containers:

- Micro Focus ALM Test Sets
- Micro Focus ALM Octane Test Suites

Artifact Creation Flow should be one way from ALM to Octane.

### Integration 2: Test Planning

The Test Planning integration is a [Work Item Synchronization](#) that flows Test Instances/Cases from ALM to Test Cases in Octane.

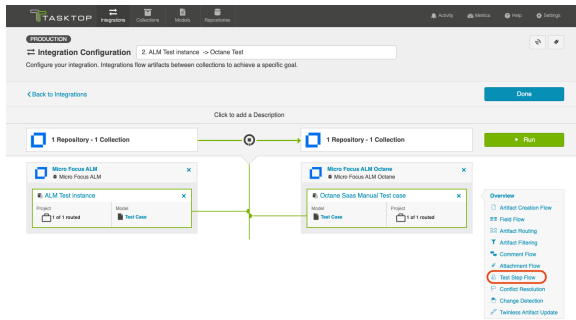


To configure this integration, you will use the normal **Work Item Synchronization** template.

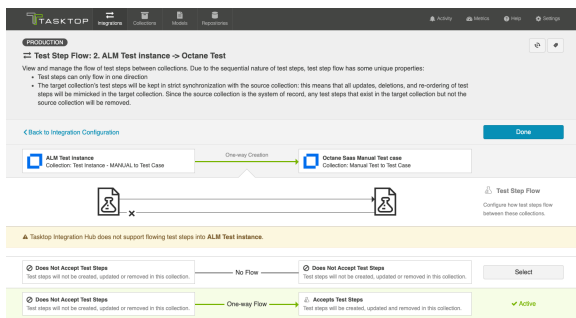
#### Work Item Collections:

- ALM Test Instances

- Configure parent relationship of ALM Test Instance to Test Set
- Octane Test Cases
  - Configure parent relationship of Octane Test to *Link to Test*



Clicking the link will bring you to the Test Step Flow screen, where you can click **Select** to choose your desired Test Step Flow style.



**i** Due to the sequential nature of test steps, test step flow has some unique properties:

- Test steps can only flow in one direction
- The target collection's test steps will be kept in strict synchronization with the source collection: this means that all updates, deletions, and re-ordering of test steps will be mimicked in the target collection. Since the source collection is the system of record, any test steps that exist in the target collection but not the source collection will be removed.
- If the test steps on the target artifact are changed by an end-user, they will be updated by Tasktop when one of the fields or ordering on the source artifact's test steps is changed.

**💡 Note:** comments and attachments are not currently supported on test steps.

### Integration 3: Test Execution

The Test Execution integration is a **Work Item Synchronization** that flows **Test Step Results** as a sub-entity located on **Octane Manual Runs** to ALM Test Runs.



**Supported Artifacts:**

- ALM Test Run
  - Configure parent relationship of ALM Test Run to *Test Instance*
- Octane Manual Run
  - Configure parent relationship of Octane Test Run to *Test*

## Jama

Use the instructions below to configure the following integrations:

- Jama Test Steps Jama Test Steps
- Jama Test Steps Octane Test Steps
- Octane Test Steps Jama Test Steps

### Step 1: Connect to your Repository

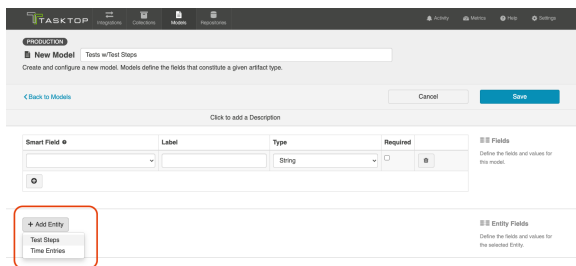
First, connect to your repository by following the instructions [here](#).

You can learn more about Jama-specific configuration in our [Connector Docs](#).

### Step 2: Construct your Model

To flow test steps, you will need to add the **Test Steps** entity when creating your Test model.

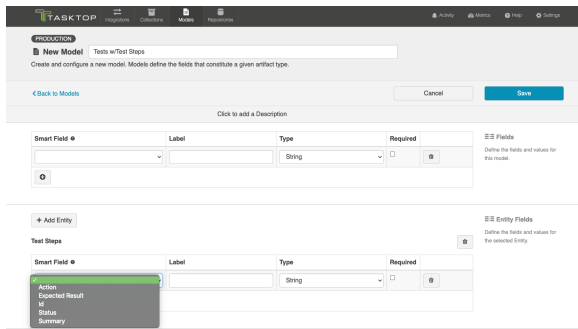
To do this, click **+ Add Entity** and select the **Test Steps** option.



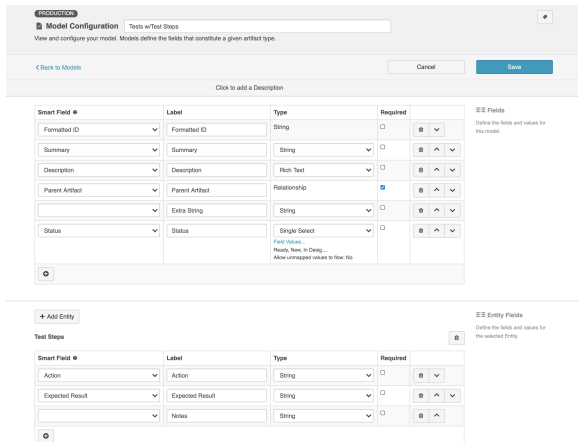
A Test Steps entity will be added. On the Model Configuration screen, you will then see two panels:

1. **Test Fields:** In the top section, add any fields you'd like to flow on the test (or test run) artifact that are not part of its associated test steps.
2. **Test Step Fields:** In the bottom section, add any fields you'd like to flow that are a part of test steps. You'll see that Tasktop provides some Smart Fields that are test step specific to help you get started, but you can add any other desired fields by leaving the Smart Field blank.

**Note:** All test steps fields must be of the **same** type (i.e., all must be string fields or all must be rich text fields), as you cannot configure different types for each test step field in Jama.



Here is an example of a very simple Test Model with Test Steps:



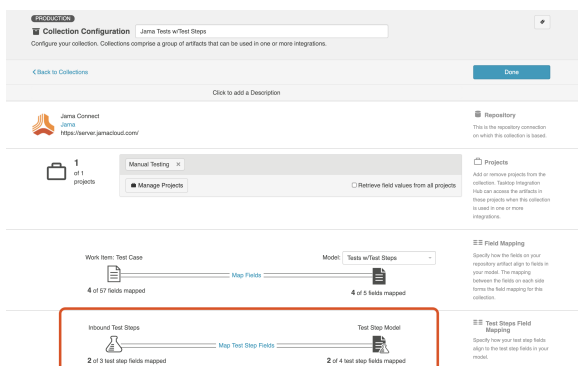
You can find general details on how to create a model [here](#).

### Step 3: Create your Collection

You can find general details on how to create a collection [here](#).

You will see a **Map Test Fields** sash on your collection if:

- Your model is a Model with Test Steps, and
- Your artifact is an Octane Test, or Jama Test Case



The process to map test step fields is very similar to the process on the normal [Field Mapping](#) screen. Note that both relationship(s) and other field types for test steps will be mapped on this one sash.



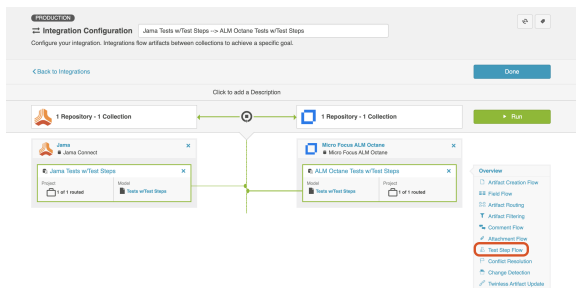
**Note:** Tests and Test Steps do not require a typical relationship field mapping to link them. We've added behind-the-scenes smarts to couple them for you.

## Step 4: Configure your Integration

You can find general details on how to configure an integration [here](#).

To see a Test Step Flow link on the Integration Configuration screen, the following conditions must be met:

- Model is of type 'Model with Test Steps'
- Artifacts in both collections have test steps
- The relevant Tasktop connectors must support test steps (see [Connector Documentation](#) to confirm)



Clicking the link will bring you to the Test Step Flow screen, where you can click **Select** to choose your desired Test Step Flow style:



**i** Due to the sequential nature of test steps, test step flow has some unique properties:

- Test steps can only flow in one direction
- The target collection's test steps will be kept in strict synchronization with the source collection. This means that all updates, deletions, and re-ordering of test steps will be mimicked in the target collection. Since the source collection is the system of record, any test steps that exist in the target collection but not the source collection will be removed.
- If the test steps on the target artifact are changed by an end-user, they will be updated by Tasktop when one of the fields or ordering on the source artifact's test steps is changed.

**Note:** Comments, attachments, and inline/embedded images are not currently supported on Test Steps.

# TestRail

Use the instructions below to configure the following integrations:

- TestRail Test Steps Jama Test Steps
- Jama Test Steps TestRail Test Steps

## Step 1: Connect to your Repository

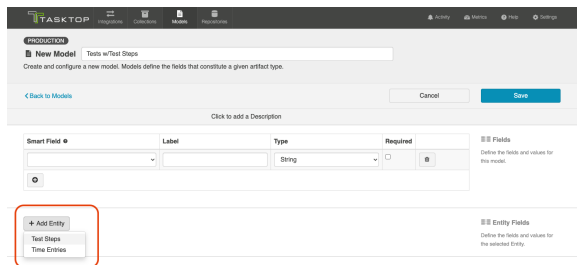
First, connect to your repository by following the instructions [here](#).

You can learn more about Jama-specific configuration in our [Connector Docs](#).

## Step 2: Construct your Model

To flow test steps, you will need to add the **Test Steps** entity when creating your Test model.

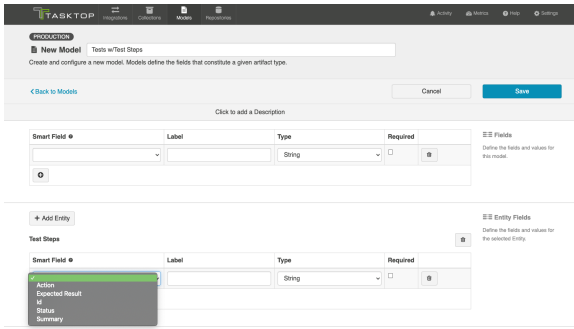
To do this, click **+ Add Entity** and select the **Test Steps** option.



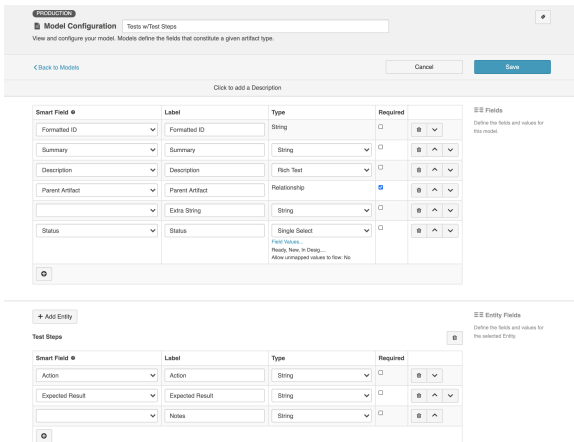
A Test Steps entity will be added. On the Model Configuration screen, you will then see two panels:

1. **Test Fields:** In the top section, add any fields you'd like to flow on the test (or test run) artifact that are not part of its associated test steps.
2. **Test Step Fields:** In the bottom section, add any fields you'd like to flow that are a part of test steps. You'll see that Tasktop provides some Smart Fields that are test step specific to help you get started, but you can add any other desired fields by leaving the Smart Field blank.

**Note:** All test steps fields must be of the **same** type (i.e., all must be string fields or all must be rich text fields), as you cannot configure different types for each test step field in Jama.



Here is an example of a very simple Test Model with Test Steps:



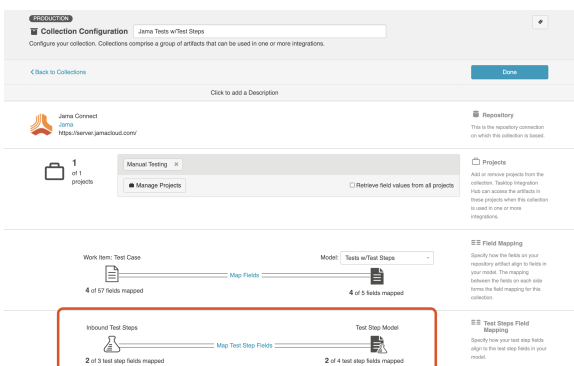
You can find general details on how to create a model [here](#).

## Step 3: Create your Collection

You can find general details on how to create a collection [here](#).

You will see a **Map Test Fields** sash on your collection if:

- Your model is a Model with Test Steps, and
- Your artifact is a Jama Test Case or TestRail Test Case



The process to map test step fields is very similar to the process on the normal [Field Mapping](#) screen. Note that both relationship(s) and other field types for test steps will be mapped on this one sash.

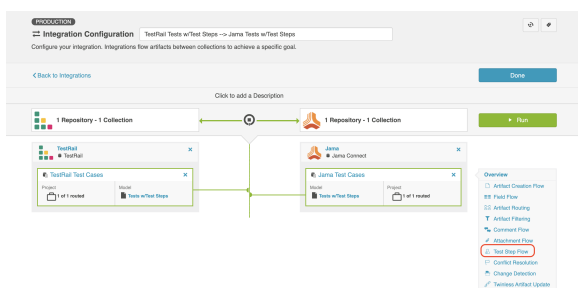
**Note:** Tests and Test Steps do not require a typical relationship field mapping to link them. We've added behind-the-scenes smarts to couple them for you.

## Step 4: Configure your Integration

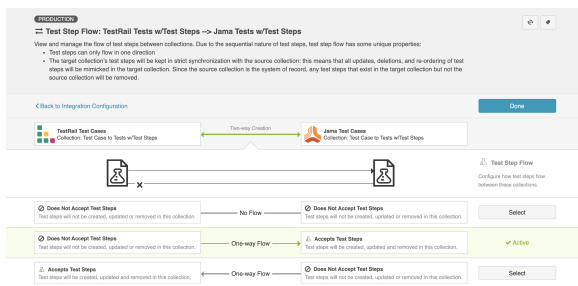
You can find general details on how to configure an integration [here](#).

To see a Test Step Flow link on the Integration Configuration screen, the following conditions must be met:

- Model is of type 'Model with Test Steps'
- Artifacts in both collections have test steps
- The relevant Tasktop connectors must support test steps (see [Connector Documentation](#) to confirm)



Clicking the link will bring you to the Test Step Flow screen, where you can click **Select** to choose your desired Test Step Flow style:



**i** Due to the sequential nature of test steps, test step flow has some unique properties:

- Test steps can only flow in one direction
- The target collection's test steps will be kept in strict synchronization with the source collection. This means that all updates, deletions, and re-ordering of test steps will be mimicked in the target collection. Since the source collection is the system of record, any test steps that exist in the target collection but not the source collection will be removed.
- If the test steps on the target artifact are changed by an end-user, they will be updated by Tasktop when one of the fields or ordering on the source artifact's test steps is changed.

# Test Result Synchronization

Test Result Synchronization is supported for the following integration scenarios:

- Tosca Test Results ALM Test Results
- ALM Test Results ALM Test Results
- ALM Octane Test Results ALM Test Results

To flow test results, please follow the instructions below.

---

## Tricentis Tosca

Many organizations have been using Micro Focus ALM (aka Quality Center) for quality management for years. But where once it was the only tool used for testing, today enterprises are augmenting ALM with additional tools to align with their agile and test automation efforts. That includes tools like Tricentis Tosca. Even as new tools are introduced, ALM remains popular and continues to play an important role in test management, especially when it comes to manual testing, defect management, and quality reporting.

The challenges for QA teams and leadership are how to restore visibility into coverage, quality, and cost, now that testing data is split across multiple tools.

Tasktop enables users to flow test results into Micro Focus ALM in order to take advantage of ALM's reporting capabilities while using other tools, such as Tricentis Tosca, for their test planning and execution.

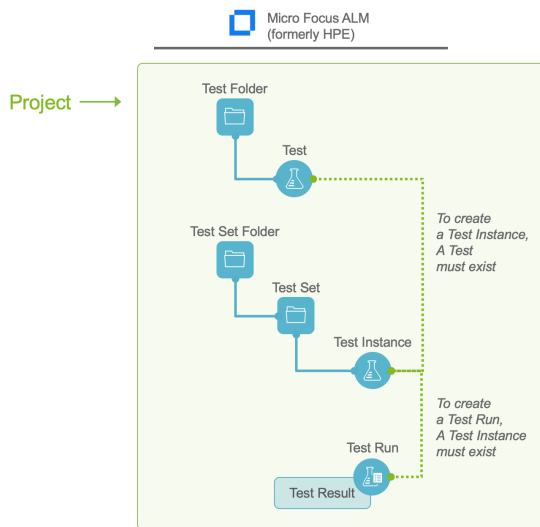
The method outlined below will enable you to flow test results into Micro Focus ALM from Tricentis Tosca or from another ALM instance. Due to the architectural specificity of each external tool, the methods below cannot be used for other endpoints.

You can watch this demo video to learn more:

## Test Architecture

Before you begin configuring your integration, it's important to understand how test artifacts relate to one another.

While the goal of this integration is to flow test results, the architecture required to do so is more complex than one might assume. Test Results are a field on Test Runs. To create a Test Run in ALM, a Test Instance must exist. For a Test Instance to exist, both a Test and a Test Set must exist (the test is 'added' to the test set and that creates a test instance). For a Test to exist, you need a Test Folder. And for a Test Instance to exist, you need a Test Set and a Test Set Folder. That's 6 different artifacts, just to flow a Test Run!

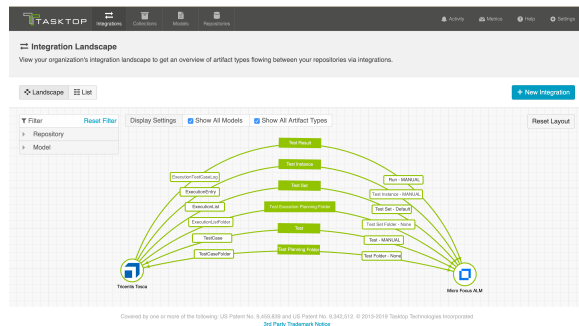


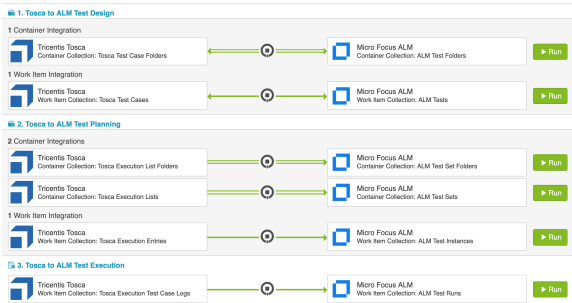
But don't worry – instead of six complex integrations, Tasktop cuts that configuration in half. To set up this integration scenario, you will set up three integrations:

Integration	Container A	Container B	Work Item
Test Design	Test Folder	--	Test
Test Planning	Test Set Folder	Test Set	Test Instance
Test Execution	--	--	Test Run

Once configured, your integrations will look like the images below.

💡 To keep your integrations in order, we recommend appending a number to the beginning of each title (i.e., "1 - Test Design," "2 - Test Planning," "3 - Test Execution").





## Before You Begin

Before you get started, you should familiarize yourself with the following steps of integration configuration:

## Step 1: Connect to your Repository

- [Standard Repository Connection](#)

## Step 2: Construct your Model

- [Create or Reuse a Model](#)

## Step 3: Create your Collection

💡 Review the details in the sections below to ensure that any required fields are mapped in your collection

- [Work Item Collection \(Repository\)](#)
- [Container Collection \(Repository\)](#)

## Step 4: Configure your Integration

- [Work Item Synchronization](#)
- [Container + Work Item Synchronization](#)

### Integration 1: Test Design

The first integration you will configure is a [Container + Work Item Synchronization](#) flowing **Test Folders /Test Case Folders** (container) and **Tests/Test Cases** (work item).



### Supported Containers:

- Micro Focus ALM Test Folders

- Parent field must be mapped to preserve folder hierarchy
- Tricentis Tosca Test Case Folders
  - Parent field must be mapped to preserve folder hierarchy


### Supported Artifacts:

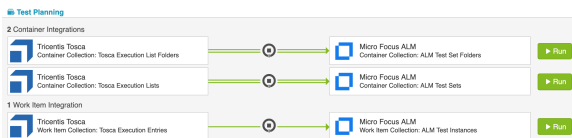
- Micro Focus ALM Tests
  - Subject field must be mapped (this points to the Test folder)
  - When flowing tests out of ALM, multiple test configurations are not supported. Tests must have a single test configuration.
- Tricentis Tosca Test Cases
  - Test Case Folder field must be mapped

This integration can be run independently, as the Test Folders and Tests do not require any other artifacts to exist before they can be created. Artifact Creation Flow can be one-way or two-way.

### Integration 2: Test Planning

The Test Planning integration is a [Container + Work Item Synchronization](#) that utilizes child containers, flowing **Test Set Folders/Execution List Folders** (container), **Test Sets/Execution Lists** (child container), and **Test Instances/Execution Entries** (Work Item).

 **Note:** You should **not** map the Status field in the Test Planning integration. The status field should **only** be mapped in the [Test Execution](#) integration.



To configure this integration, you will use the normal 'Container + Work Item Synchronization' template. Tasktop has behind-the-scenes magic that will allow you to include a child-container integration once it sees the appropriate collections created:

### Container Collections:

- ALM Test Set Folders
  - Parent field must be mapped to preserve folder hierarchy
- Tosca Execution List Folders
  - Parent field must be mapped to preserve folder hierarchy

### Child Container Collections:

- ALM Test Sets
  - Parent field must be mapped to preserve folder hierarchy
- Tosca Execution Lists
  - Parent field must be mapped to preserve folder hierarchy

### Work Item Collections:



- ALM Test Instances
  - Test field must be mapped
  - Test Set field must be mapped
  - Status field must **not** be mapped
- Tosca Execution Entries
  - Test Case field must be mapped
  - Execution List field must be mapped
  - Status field must **not** be mapped

### Step 1: Test Set Folder/Execution List Folders

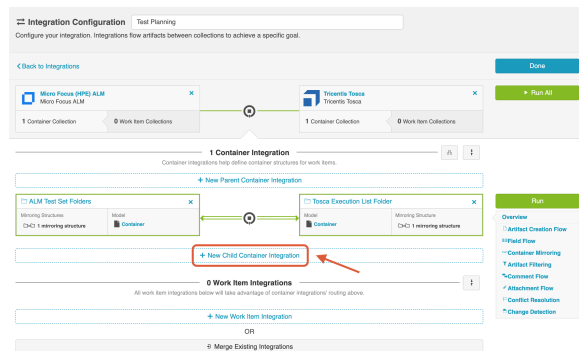
Once your collections have been created and configured, create a [Container + Work Item Synchronization](#) . First, configure the top-level container integration:

- ALM <=> ALM: Test Set Folder to Test Set Folder, or
- Tosca <=> ALM: Execution List Folder to Test Set Folder

Container Creation Flow will most likely be one way into ALM, but this will depend on the use case.

Once this integration has been configured, you'll see an option to create a **child container integration**.

**Note:** The Test Instance (or Execution Entry) collection (i.e., the work item collection for this integration) **must** exist before the "New Child Container Integration" button will appear while configuring this integration.



### Step 2: Test Sets/Execution Lists

Your Child Container Integration will be

- ALM <=> ALM: Test Set to Test Set, or
- Tosca <=> ALM: Execution List to Test Set

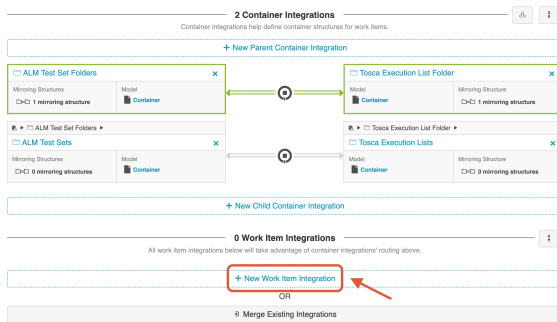
This integration will likely not require container mirroring configuration, as it will inherit that from the parent container integration.

### Step 3: Test Instances/Execution Entry

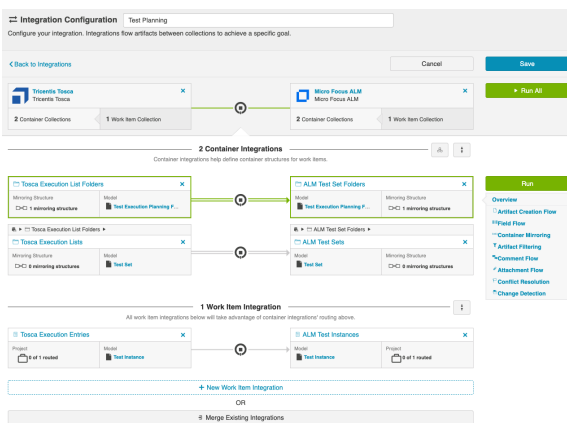
Finally, you will configure your Work Item integration. It will either be:

- ALM <=> ALM: Test Instance to Test Instance
- Tosca <=> ALM: Execution Entry to Test Instance

Click **New Work Item Integration** to add the integration.



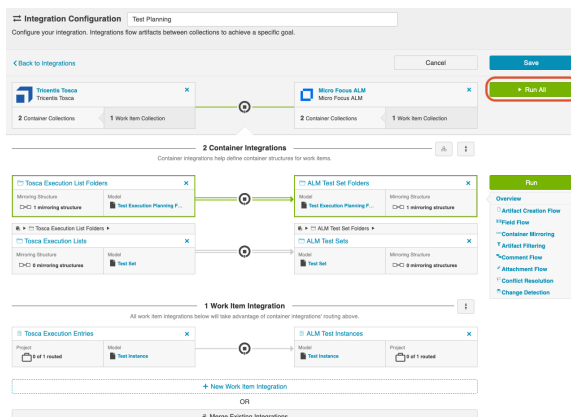
Here is what your fully configured integration will look like:



## Step 4: Run Integration

💡 Before you run this integration, you must have the [Test Design integration](#) configured and running. This is because in order to create a Test Instance, both a Test Set *and* a Test (created via the Test Design integration) must exist. When a Test is added to a Test Set, a Test Instance is created.

To run the integration, click the green **Run All** button.



## Integration 3: Test Execution

The Test Execution integration is a [Work Item Synchronization](#) that flows **Test results** located on **ALM Test Runs** or **Tosca Execution Test Case Logs**.



### Supported Artifacts:

- ALM Test Run
  - Test Instance field must be mapped
  - Status field must be mapped
- Tosca Execution Test Case Log
  - Execution Entry field must be mapped
  - Status field must be mapped

### Artifact Routing and Filtering

Because Test Runs and Execution Test Case Logs live at the project level, Artifact Routing will only need to be configured at the Project level.

Since routing is at the project level, you may be asking, "How will I know that only the Test Runs that I care about are synchronizing? I don't want every single Test Run in this project to flow!" And you're in luck. Test Execution Integrations behave a little differently from typical integrations: Tasktop will use built-in magic to *only* flow the Test Runs or Execution Test Case Logs that are associated with a Test Instance/Execution Entry *that is also configured to flow*.

It's that simple!

For this reason, you will see an Issue on the [Activity Screen](#) of Tasktop if you attempt to run this integration without running an associated Test Planning integration (Remember: Test Runs require that an associated Test Instance exist first).

### Race Conditions

Due to the interdependencies between the three integrations, the order that artifacts synchronize in matters.

In this Test Management integration scenario, Tasktop won't create an artifact if its parent container or other required artifact does not yet exist.

Examples of when Tasktop won't flow a work item:

- Trying to create a Test Run without the correct Test Instance already existing in the target system
- Trying to create a Test Instance without the correct Test already existing in the target system

Here's an example of what you can expect to see in a race condition:

- Create a test instance and immediately run the test
- If Tasktop picks up the Test Run first, it will not have the necessary Test Instance on the target to attach to and will error
- Once Tasktop picks up the Test Instance & synchronizes it, then the Test Run will be able to flow across on retry and the error will clear

You are most likely to see this condition when first setting up your integrations. For this reason, we recommend setting up the integrations 'from top to bottom'. In other words, start with the Test Design integration. Then move on to the Test Planning integration. And finally, set up the Test Execution integration. If you have the integrations running in that order, you'll be more likely to flow any required artifacts *before* any dependent artifacts attempt to flow.

💡 To keep your integrations in order in the Integration List View, we recommend appending a number to the beginning of each title, i.e. "1 - Test Design," "2 - Test Planning," "3 - Test Execution"

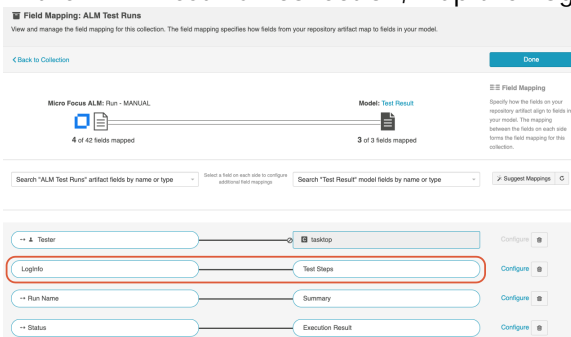
Another possible time when this race condition could occur is if you have vastly different change detection intervals on your integrations. For example, if you have a short interval on your Test Execution integration, but a much longer interval on your Test Planning integration.

## Flowing Test Step Results in Tricentis Tosca

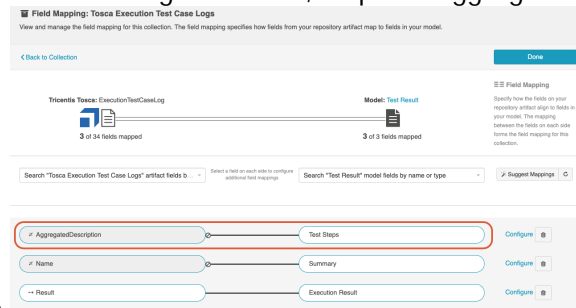
Because test steps are represented as a field on the Execution Test Case Log in Tosca, rather than as a unique child artifact (as they are in ALM), if you would like to flow test steps from ALM Tosca or Tosca ALM, it must be configured as part of a Test Execution integration.

To configure test step flow for Tosca, follow the instructions below:

1. Create a rich text field on the model used in the Test Execution integration. For explanatory purposes, we'll call this the "Test Steps" field in the model. Since test steps are represented as a field, rather than a separate artifact, your model type will be Standard Model.
2. Within ALM, create a custom text field on the ALM Test Run artifact. This field will accept the combined results of all the associated test steps from Tosca. For explanatory purposes, let's call this the LogInfo field.
3. In the ALM Test Run collection, map the LogInfo field to the Test Steps field in the model.

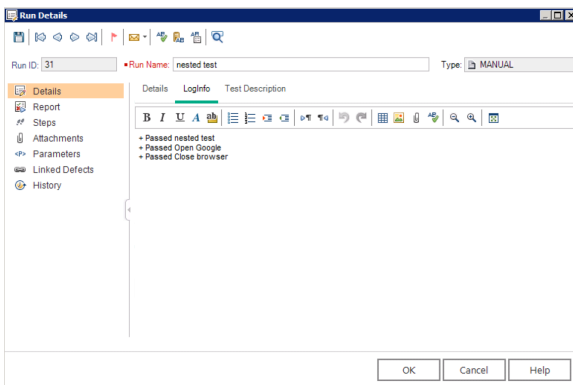


4. In the Tosca Execution Test Case Log collection, map the AggregatedDescription field to the Test



Steps field in the model.

5. Once the integration is run, you'll see that the results of each individual step in Tosca are now populated in the new custom LogInfo field on the ALM Test Run.



## Micro Focus ALM

Many organizations have been using Micro Focus ALM (aka Quality Center) for quality management for years. ALM remains popular and continues to play an important role in test management, especially when it comes to manual testing, defect management, and quality reporting.

The challenges for QA teams and leadership are how to restore visibility into coverage, quality, and cost, now that testing data is split across multiple instances.

The method outlined below will enable you to flow test results into Micro Focus ALM from another ALM instance. Due to the architectural specificity of ALM, the methods below cannot be used for other endpoints.

You can watch this demo video to learn more:

### Step 1: Connect to your Repository

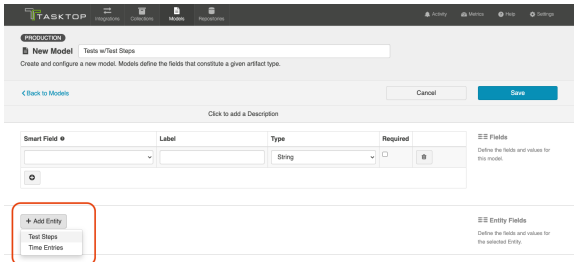
First, connect to your repository by following the instructions [here](#).

You can learn more about ALM-specific configuration in our [Connector Docs](#).

## Step 2: Construct your Model

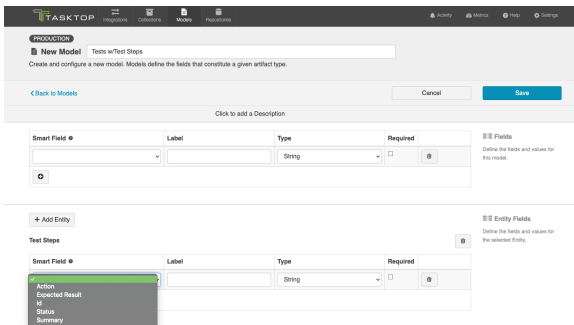
To flow test steps, you will need to add the **Test Steps** entity when creating your Test model.

To do this, click **+ Add Entity** and select the **Test Steps** option.

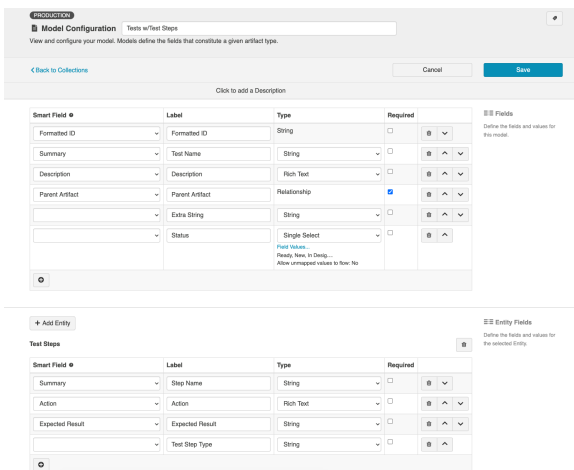


A **Test Steps** entity will be added. On the Model Configuration screen, you will now see two panels:

1. **Test Fields:** In the top section, add any fields you'd like to flow on the test (or test run) artifact that are not part of its associated test steps.
2. **Test Step Fields:** In the bottom section, add any fields you'd like to flow that are a part of test steps. You'll see that Tasktop provides some Smart Fields that are test step specific to help you get started, but you can add any other desired fields by leaving the Smart Field blank.



Here is an example of a very simple Test Model with Test Steps:



You can find general details on how to create a model [here](#).

## Converting Model Types

Standard Models can be converted to Test Models simply by choosing **Model with Test Steps** in the Model Type drop down.

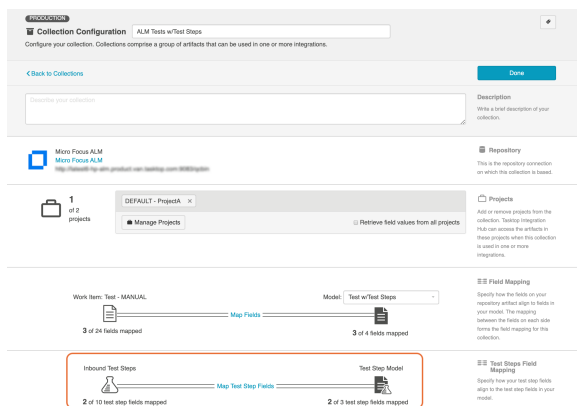
Models with Test Steps cannot be converted back to Standard Models, but this should not present any challenges to integration scenarios. If Test Step Flow is not enabled in the related integrations, the test step fields in the model will simply be ignored.

## Step 3: Create your Collection

You can find general details on how to create a collection [here](#).

You will see a **Map Test Fields** sash on your collection if:

- Your model is a Model with Test Steps, and
- Your artifact is an ALM Test, ALM Test Run, Octane Test, or Octane Manual Run



The process to map test step fields is very similar to the process on the normal [Field Mapping](#) screen. Note that both relationship(s) and other field types for test steps will be mapped on this one sash.

💡 Tests and Test Steps do not require a typical relationship field mapping to link them. We've added behind-the-scenes smarts to couple them for you.

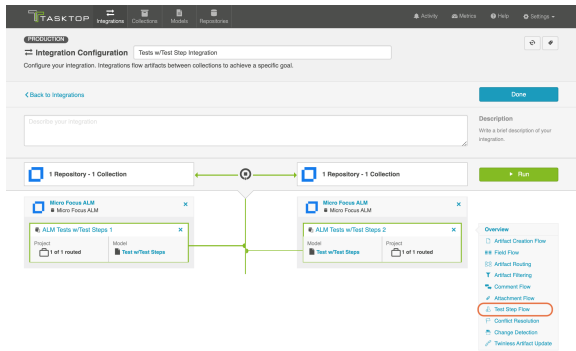
## Step 4: Configure your Integration

You can find general details on how to configure an integration [here](#).

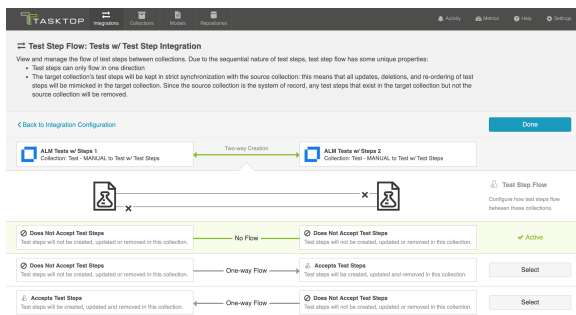
In order to see a Test Step Flow link on the integration configuration screen, the following conditions must be met:

- Model is of type 'Model with Test Steps'
- Artifacts in both collections have test steps

- The relevant Tasktop connectors must support test steps (see [Connector Documentation](#) to confirm)



Clicking the link will bring you to the Test Step Flow screen, where you can click **Select** to choose your desired Test Step Flow style.



**i** Due to the sequential nature of test steps, test step flow has some unique properties:

- Test steps can only flow in one direction
- The target collection's test steps will be kept in strict synchronization with the source collection: this means that all updates, deletions, and re-ordering of test steps will be mimicked in the target collection. Since the source collection is the system of record, any test steps that exist in the target collection but not the source collection will be removed.
- If the test steps on the target artifact are changed by an end-user, they will be updated by Tasktop when one of the fields or ordering on the source artifact's test steps is changed.

**Note:** Comments and attachments are not currently supported on test steps.

## Micro Focus ALM Octane

Many organizations have been using Micro Focus ALM (aka Quality Center) for quality management for years. But where once it was the only tool used for testing, today enterprises are augmenting ALM with additional tools to align with their agile and test automation efforts. That includes tools like Micro Focus ALM Octane.



The challenges for QA teams and leadership are how to restore visibility into coverage, quality, and cost, now that testing data is split across multiple tools.

Tasktop enables users to flow test results into Micro Focus ALM in order to take advantage of ALM's reporting capabilities while using other tools, such as ALM Octane, for their test planning and execution.

The method outlined below will enable you to flow test results into Micro Focus ALM from ALM Octane. Due to the architectural specificity of each external tool, the methods below cannot be used for other endpoints.

You can watch this demo video to learn more:

## Test Architecture

Before you begin configuring your integration, it's important to understand how test artifacts relate to one another.

While the goal of this integration is to flow test results, the architecture required to do so is more complex than one might assume.

In this integration scenario, the challenge is synchronizing test steps between the two tools:

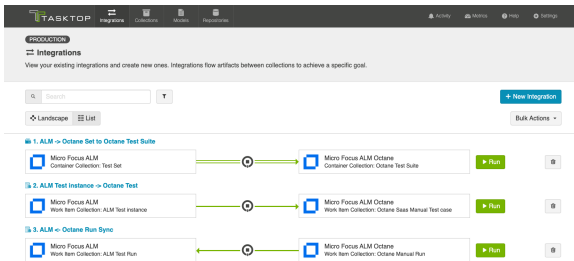
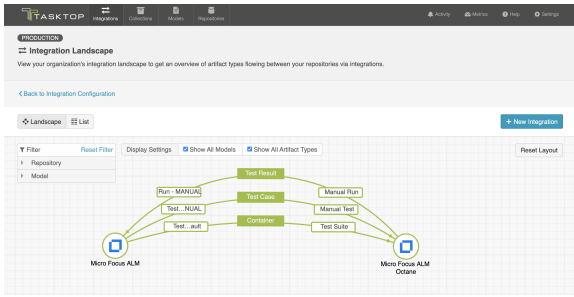
- In ALM, **Test Steps** are sub-entities of the **Test** artifact and referenced by a specific **Test Instance**.
- In ALM Octane, **Test Steps** are fields on the **Test** artifact.

The goal is to synchronize the ALM **Test Instance** artifact with the ALM Octane **Test** artifact, while including the **Test Steps** that exist on the related ALM **Test** artifact.

But don't worry - it only requires three integrations:

Integration	Container A	Container B	Work Item
Test Design	Test Set	Test Suite	--
Test Planning	--	--	Test Instance
Test Execution	--	--	Test Run

Once configured, your integrations will look like the images below.



**Tip:** To keep your integrations in order, we recommend appending a number to the beginning of each title (i.e., "1 - Test Design," "2 - Test Planning," "3 - Test Execution").

Now that you've familiarized yourself with the test architecture for this integration scenario, let's get started!

## Step 1: Connect to Your Repository

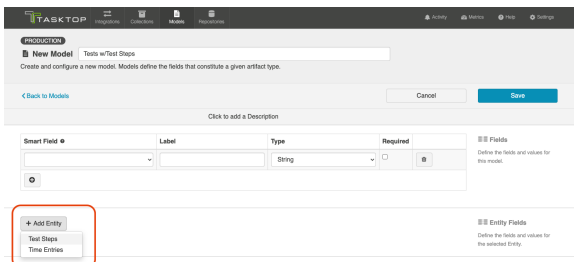
First, connect to your repository by following the instructions [here](#).

You can learn more about ALM-Octane specific configuration in our [Connector Docs](#).

## Step 2: Construct your Model

To flow test steps, you will need to add a **Test Steps** entity when creating your Test model.

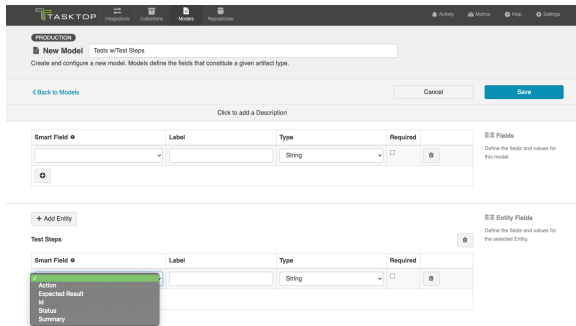
To do this, click **+ Add Entity** and select the **Test Steps** option.



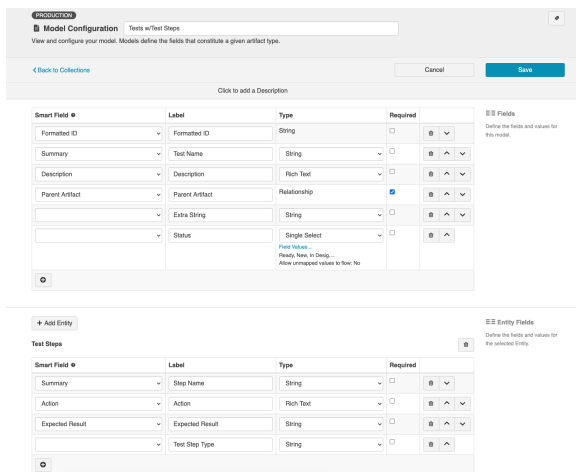
A **Test Steps** entity will be added. On the Model Configuration screen, you will then see two panels:

1. **Test Fields:** In the top section, add any fields you'd like to flow on the test (or test run) artifact that are not part of its associated test steps.

2. **Test Step Fields:** In the bottom section, add any fields you'd like to flow that are a part of test steps. You'll see that Tasktop provides some Smart Fields that are test step specific to help you get started, but you can add any other desired fields by leaving the Smart Field blank.



Here is an example of a very simple Test Model with Test Steps:



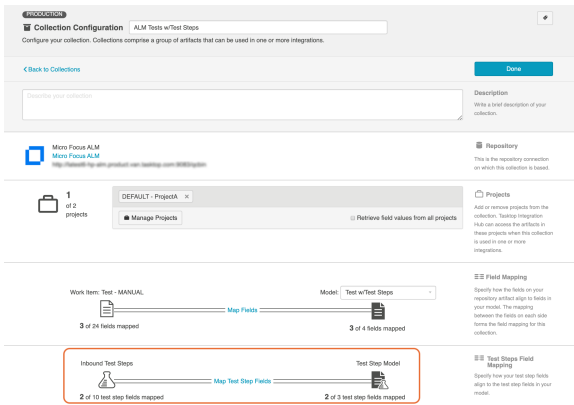
You can find general details on how to create a model [here](#).

## Step 3: Create your Collection

You can find general details on how to create a collection [here](#).

You will see a **Map Test Fields** sash on your collection if:

- Your model is a Model with Test Steps, and
- Your artifact is an ALM Test or Octane Test



The process to map test step fields is very similar to the process on the normal [Field Mapping](#) screen. Note that both relationship(s) and other field types for test steps will be mapped on this one sash.

💡 Tests and Test Steps do not require a typical relationship field mapping to link them. We've added behind-the-scenes smarts to couple them for you.

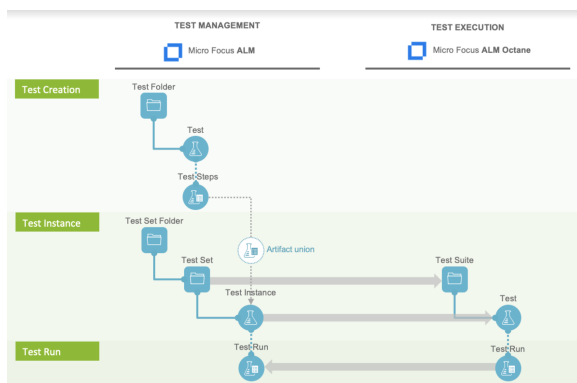
## Step 4: Configure an Artifact Union

To flow Test Steps from ALM to Octane, you will need to configure an artifact union between ALM Test Instances and ALM Tests so that the Test Steps will flow into the Octane Tests collection.

Artifact unions specify the related artifacts whose fields may flow along with the artifacts in your collection.

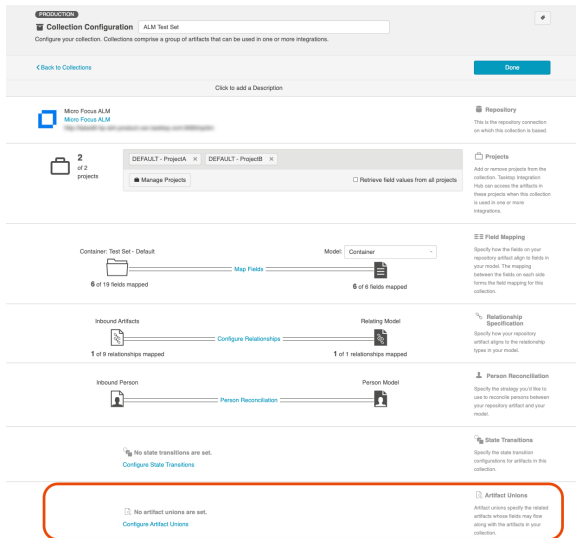
As ALM Test Steps only exist on the ALM Test artifact, test steps cannot flow from ALM to Octane. With the help of an artifact union, through a shared relationship field between ALM Test Instances and ALM Tests, the test steps are able to flow.

Learn more about artifact unions [here](#).

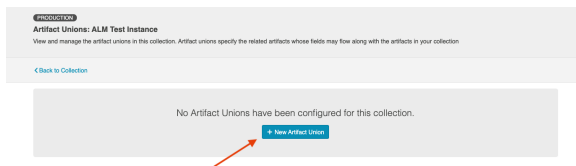


To access artifact unions, navigate to the Collection Configuration screen and click **Configure Artifact Unions**.

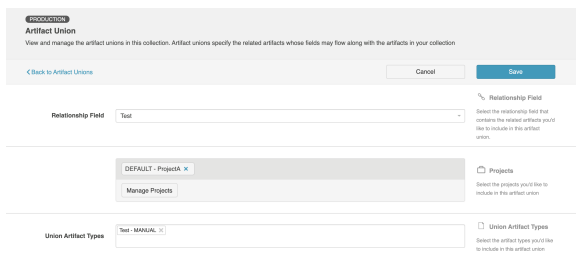
💡 The Artifact Unions sash will only be visible when there is a single relationship/container field available for the collection.



Navigate to the ALM Test Instance collection and create a new artifact union.



Then, select the **Test** relationship field, desired project(s), and artifact type.



After saving the artifact union, click **Map Test Step Fields** on the Collection Configuration screen.

On this screen, select the **Test** artifact union from the **Flow Test Steps From** dropdown menu.

After you've selected where you'd like the test steps to flow from, click **Save**.

The next step will be to configure the necessary integrations.

## Step 5: Configure your Integrations

You can find general details on how to configure an integration [here](#).

In order to see a Test Step Flow link on the integration configuration screen, the following conditions must be met:

- Model is of type 'Model with Test Steps'
- Artifacts in both collections have test steps
- The relevant Tasktop connectors must support test steps (see [Connector Documentation](#) to confirm)

## Integration 1: Test Design

The first integration you will configure is a [Container + Work Item Synchronization](#) flowing **Test Sets /Test Suites** (container).



### Supported Containers:

- Micro Focus ALM Test Sets
- Micro Focus ALM Octane Test Suites

Artifact Creation Flow should be one way from ALM to Octane.

## Integration 2: Test Planning

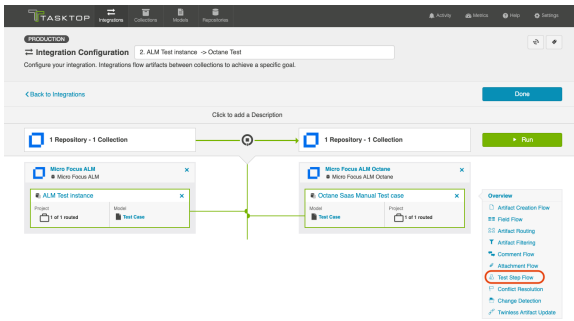
The Test Planning integration is a [Work Item Synchronization](#) that flows Test Instances/Cases from ALM to Test Cases in Octane.



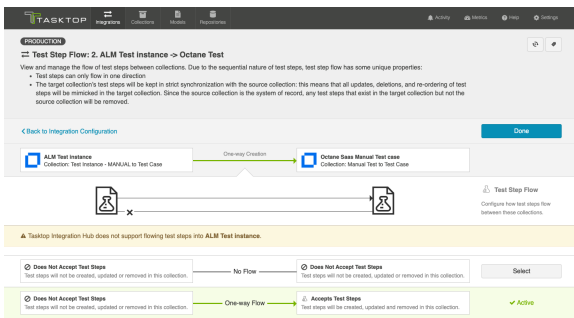
To configure this integration, you will use the normal 'Work Item Synchronization' template.

### Work Item Collections:

- ALM Test Instances
  - Configure parent relationship of ALM Test Instance to *Test Set*
- Octane Test Cases
  - Configure parent relationship of Octane Test to *Link to Test*



Clicking the link will bring you to the Test Step Flow screen, where you can click **Select** to choose your desired Test Step Flow style:



**i** Due to the sequential nature of test steps, test step flow has some unique properties:

- Test steps can only flow in one direction
- The target collection's test steps will be kept in strict synchronization with the source collection: this means that all updates, deletions, and re-ordering of test steps will be mimicked in the target collection. Since the source collection is the system of record, any test steps that exist in the target collection but not the source collection will be removed.
- If the test steps on the target artifact are changed by an end-user, they will be updated by Tasktop when one of the fields or ordering on the source artifact's test steps is changed.

**Note:** Comments and attachments are not currently supported on test steps.

## Integration 3: Test Execution

The Test Execution integration is a **Work Item Synchronization** that flows **Test Step Results** as a sub-entity located on **Octane Manual Runs** to ALM Test Runs.



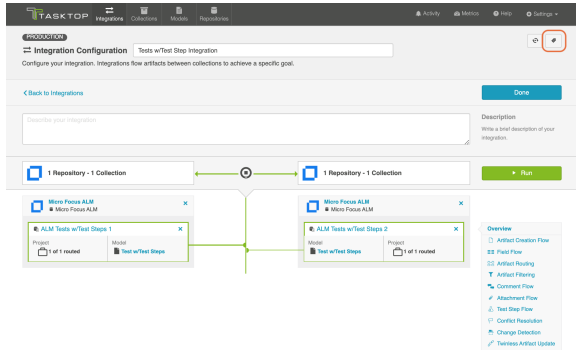
### Supported Artifacts:

- ALM Test Run
  - Configure parent relationship of ALM Test Run to *Test Instance*
- Octane Manual Run

- Configure parent relationship of Octane Test Run to *Test*

## Viewing Associated Configuration Elements

To view associated configuration elements (such as collections or models that utilize the Test Step Synchronization or Test Result Synchronization you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.



### Associated Elements for Integration "Tests w/Test Step Integration"

- ▣ **2 Repository Collections used by this Integration**
  - [ALM Tests w/Test Steps 1](#)
  - [ALM Tests w/Test Steps 2](#)
- ▣ **1 Model used by this Integration**
  - [Test w/Test Steps](#)
- ▣ **1 Repository Connection used by this Integration**
  - [Micro Focus ALM](#)

Close



# Step 5: Expand or Modify your Integration

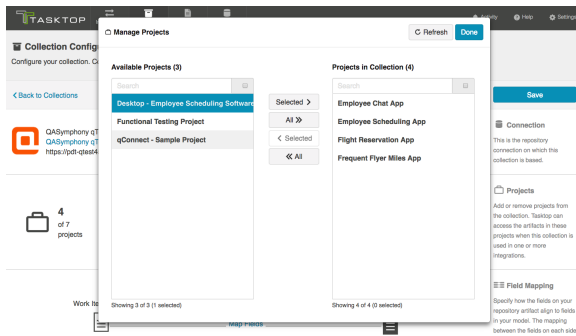
## Expanding the Scale of Your Integration

You've already configured your integration, and it's running great! Now you'd like to increase the scale by adding additional projects from each of your repositories to your integration landscape, or by configuring additional field mappings. No problem - you can make these updates in just a few clicks!

Below, we've included some tips and tricks on how to effectively scale your integration, as well as information on what to expect when you make modifications to your integration configuration after the integration has been activated.

## Adding Projects

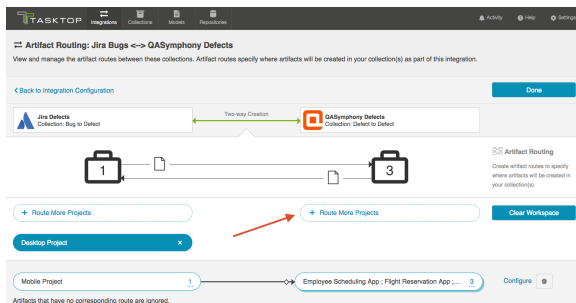
In order to add additional projects from one or more of your repositories to your integration landscape, simply navigate to each [collection](#), and add additional projects as desired. If you don't see a project you'd like to add, click the 'refresh' button in the upper right corner.



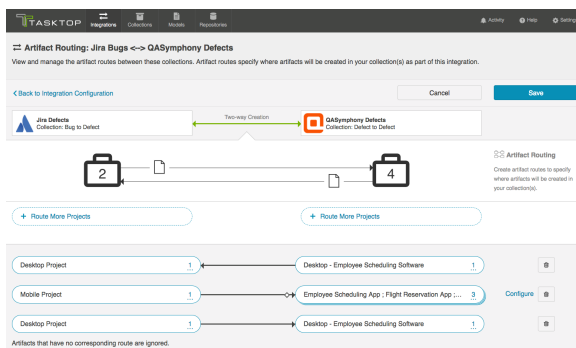
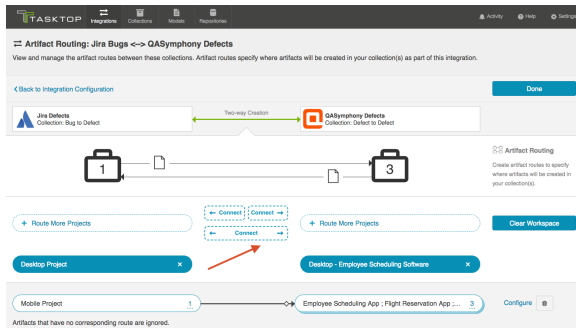
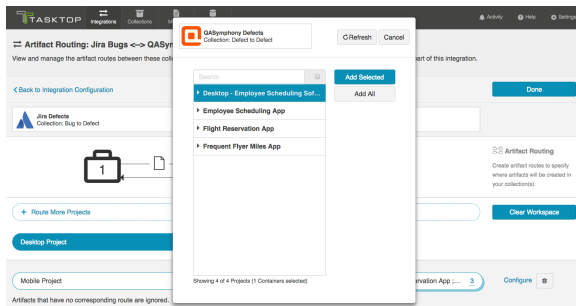
Once your updated projects are saved, navigate to the integration, click on [Artifact Routing](#) and route the projects as desired - either creating new routes or adding to existing routes (see instructions below).

Once the new projects have been added and routed, Tasktop will detect the artifacts contained within the new project(s) at the [change detection interval](#) and flow data according to the configuration that you have already set.

Add Projects to New Routes:

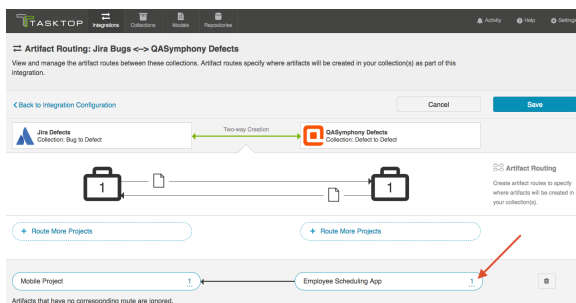


💡 Note: If you don't see desired projects, click the 'Refresh' button in the upper right corner.



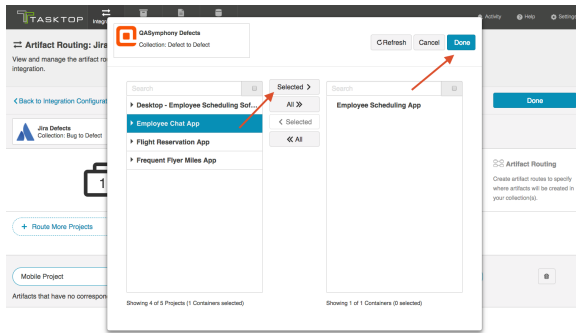
## Add Projects to Existing Routes:

To add additional projects to an existing route, click the numerical link on the right side of the pill.

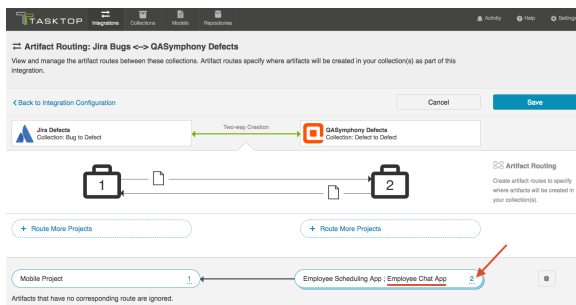


Highlight the project you'd like to add, click 'Selected>' and then 'Done.'

💡 Note: If you don't see the project you'd like to add, click the 'Refresh' button in the upper right corner.



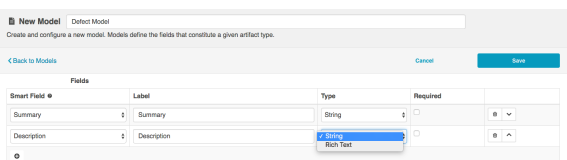
You will now see the updated number of projects, and the additional project's name listed in the pill:



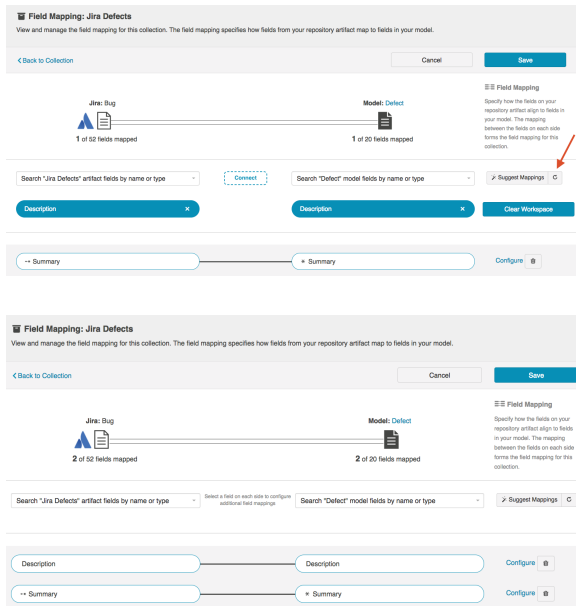
💡 Note: Depending on how you set up your artifact routing, you may need to configure conditional artifact routing. This will be relevant if you route to more than one target project (as you will need to identify criteria by which the integration can determine which project to flow the artifact to). You can learn more about conditional artifact routing [here](#). If you'd like to set up conditional routes based on a field on the artifact that is not yet part of your model, see details in the section below to learn how to add that new field.

## Adding or Editing Fields

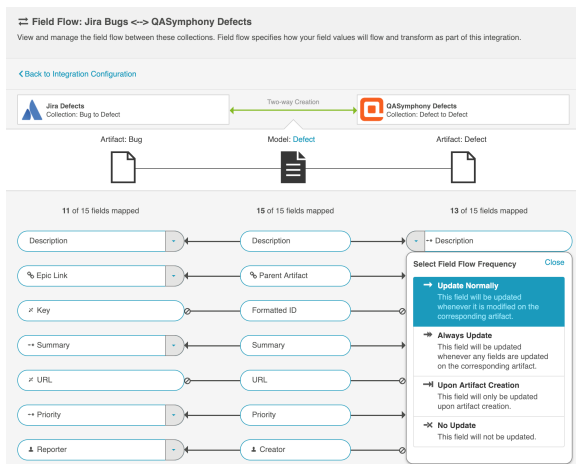
If you'd like to add, remove, or change a field mapping, Tasktop allows you to do so even after the integration has been run. To add a new field, first make sure it's accounted for in your model. If needed, you can add a new field on the [Model screen](#):



Once the field has been added to your model, navigate to your relevant [collections](#) and [map that field](#) as needed. If you don't see the field listed, click the 'refresh' button next to 'Suggest Mappings.'



You can then edit the field flow frequency from the integration's [field flow](#) screen.



If you add a new field to your integration's field flow, the field will be synced automatically for **newly created artifacts**, as detected based on the [change detection interval](#).

**⚠** Note that if you edit or add a new field mapping to an integration that is already running, Tasktop will **not** automatically apply the new field mapping to artifacts that had already been synced and that were created before that mapping was added until those existing artifacts are picked up by change detection. This means that the new field, or another qualifying field, must change on the artifact before Tasktop will update the new field. If needed, you can use the 'Process All Artifacts' button to force updates through to that field. Please review section [below](#) before using that feature.

## Updating Routes

There may be rare scenarios when you must move routes from an existing integration to another integration. For example, you may have configured Tasktop incorrectly and mistakenly created multiple integrations which should be combined into one. Or you could be upgrading Micro Focus

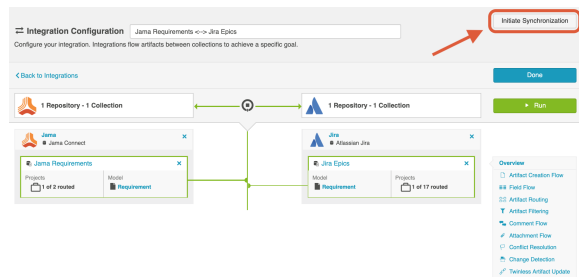
(HPE) ALM, which requires moving projects from an instance running an old version of ALM to a server running a newer instance of ALM. To learn how to move routes from one integration to another, see [here](#).

Since modifications made to existing routes in a running integration can impact internal artifact associations, please contact [Tasktop Support](#) before making such modifications.

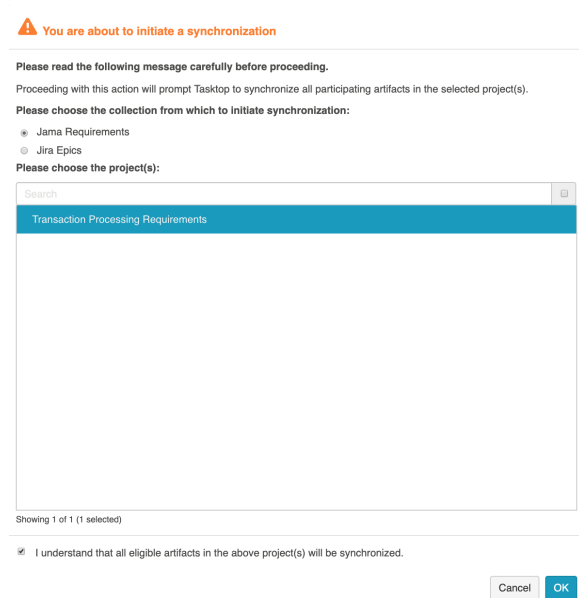
## Expanding Artifact Filters

If you update an Artifact Filter of a running integration so that it includes additional artifacts, you can choose to initiate a synchronization immediately in order to synchronize the newly eligible artifacts.

To initiate synchronization, go to the main integration configuration screen and click **Initiate Synchronization** in the upper right corner.



On the pop-up that appears, select the collection and project(s) whose artifacts you'd like to synchronize:



This will immediately trigger a special high fidelity full scan for the project(s) selected, causing eligible artifacts in those project(s) to synchronize.

# Changing Repository URL

If you need to change the location for an existing repository that is already part of a running integration, we recommend contacting [Tasktop Support](#) to prevent disruptions to existing integrations. As a general rule, we do not recommend creating a new repository connection to replace the repository for an existing integration.

If you are upgrading Micro Focus (HPE) ALM, please review how to move routes between existing integrations [here](#).

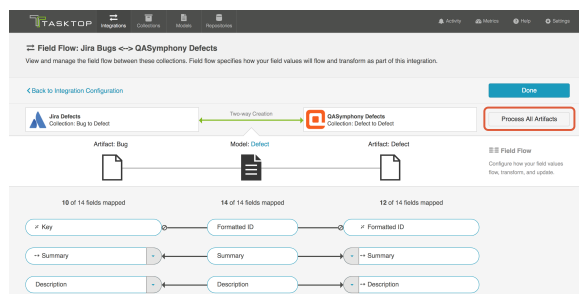
## Processing All Artifacts

**⚠️** Please contact [Tasktop Support](#) before using this feature to ensure you understand its impacts.

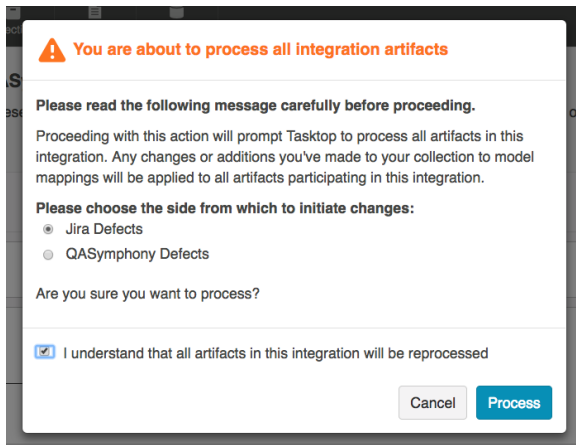
If you'd like force through updates for all artifacts in a collection of your integration, you can click the 'process all artifacts' button on the Field Flow screen. This can be useful if you add a new field mapping to your configuration or if you change your artifact routing or artifact filtering criteria to add new artifacts to your integration.

When 'Process All Artifacts' is clicked, Tasktop forces an extra special [High Fidelity Full Scan](#) to run at the next change detection interval. Unlike a typical High Fidelity Full Scan, this will scan ALL artifacts within the collection (regardless of whether they have synchronized or not) and also mark all artifacts as changed. This means that it will pick up artifacts that are newly eligible for the integration based on updated routing or filtering as well as process newly configured field mappings. As such, users should expect that this feature can lead to high server load on the external repositories.

If you are planning to use the 'Process All Artifacts' button to force updates for a new field, make the field flow for the new field one-way from the desired source collection, with 'update always' frequency. This will help ensure that the initial population is done completely. Once all artifacts have been processed, change the field flow direction and frequency to your desired configuration.



After clicking 'Process All Artifacts,' you will be prompted to choose the side from which to initiate changes:



This will process all artifacts in the source collection upon the next change detection interval, and flow any eligible field updates to the target collection.

# Resources

## Help and Support

To learn more about Tasktop, see [our website](#).

For help, contact us at the [Tasktop Support Center](#).

## Feedback and Ideas

Have a suggestion or an idea for the product? Please contact us at [feedback@tasktop.com](mailto:feedback@tasktop.com).



# PDF Download

You can download a PDF of our Quick-Start Guide here: [Tasktop Integration Hub - 22.3 Quick Start Guide.pdf](#)




You can download a PDF of our Installation and Maintenance Guide here: [Tasktop Integration Hub - 22.3 Installation and Maintenance Guide.pdf](#)

# Supported Repository Versions






## Tasktop Integration Hub 22.3 (July 19, 2022)




✔ If you are interested in extended support, please reach out to your [Tasktop contact](#) before the date specified [here](#).

⚠ Tasktop Integration Hub Cloud can only connect to On-prem repositories if customers [allow such network connections through their firewall](#).

	Repository	General Support (Tasktop 22.3)
	Aras Innovator	11.0 SP15
	Asana	Current On Demand (Cloud) Version
	Atlassian Jira Core	8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 9.0  Support All Patch Versions  Current On Demand (Cloud) Version
	Atlassian Jira Software	8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17,

		<p>8.18, 8.19, 8.20, 8.21, 8.22, 9.0</p> <p>Support All Patch Versions</p> <p>Current On Demand (Cloud) Version</p>
	<p>Xray Test Management for Jira</p>	<p>6.2.1</p>
	<p>Atlassian Jira Service Management</p>	<p>4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 5.0</p> <p>Support All Patch Versions</p>
	<p>Atlassian Jira Align</p>	<p>Current On Demand (Cloud) Version</p>
	<p>Blueprint</p>	<p>10.0, 10.1, 10.2, 10.3, 11.0, 11.1, 12.0, 12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7, 13.1, 13.2</p> <p>4.0, 4.1, 4.2, 5.0, 5.1 (Storyteller)</p> <p>Support All Patch Versions</p> <p>Current On Demand (Cloud) Version</p> <p><b>Note:</b> With the release of Blueprint version 11.2, Blueprint Storyteller has been rebranded as Blueprint. Blueprint Storyteller versions 4.0 - 5.1 remain supported.</p>
	<p>BMC Remedy</p>	

		<p>9.1.03, 9.1.04, 18.05, 18.08, 19.02, 19.08, 19.11, 20.02</p> <p>Support All Patch Versions</p>
	<p>Broadcom Clarity</p>	<p>15.7, 15.7.1, 15.8, 15.8.1, 15.9, 15.9.1, 16.0.0</p> <p>Support All Patch Versions</p> <p>Current On Demand (Cloud) Version</p>
	<p>Broadcom Rally</p>	<p>2018.1, 2.0, 2.1.0</p> <p>Support All Patch Versions</p> <p>Current On Demand (Cloud) Version</p>
	<p>Cherwell Service Management</p> <p>(Only available for Planview OEM)</p>	<p>10.0.2</p>
	<p>codebeamer</p>	<p>8.2, 9.0, 9.1, 9.2, 9.3, 9.4, 9.5, 10.0, 10.1, 20.11, 21.04, 21.09</p> <p>Support All Patch Versions</p>
	<p>Digital.ai Agility</p>	<p>Enterprise and Ultimate: 21.0, 21.1, 21.2, 21.3, 22.0, 22.1</p> <p>Support All Patch Versions</p> <p>Current On Demand (Cloud) Version</p>

	Digital.ai Release	8.0, 8.2, 8.5, 9.8, 10.0, 10.1, 10.2, 10.3
	Git	<p>All</p> <p><b>*Note:</b> If using a supported Git Hosting Service, the version of the service used does not impact functionality. It is used to determine commit location.</p> <p>⚠ Git connector is not available on Tasktop Cloud</p>
	GitHub Issues	<p>Enterprise 3.0 and prior, 3.1, 3.2, 3.3, 3.4</p> <p>Current On Demand (Cloud) Version</p>
	GitLab Issues	<p>Enterprise and Community Edition:</p> <p>12.0, 12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7, 12.8, 12.9, 12.10, 13.0, 13.1, 13.2, 13.3, 13.4, 13.5, 13.6, 13.7, 13.8, 13.9, 13.10, 13.11, 13.12, 14.0, 14.1, 14.2, 14.3, 14.4, 14.5, 14.6, 14.7, 14.8, 14.9</p> <p>Support All Patch Versions</p> <p>Current On Demand (Cloud) Version</p>
	IBM Rational ClearQuest	9.1.0
	IBM Engineering Requirements Management DOORS	9.5, 9.5.2, 9.6, 9.6.1, 9.7, 9.7.1, 9.7.2, 9.7.2.4





Family	⚠ IBM DOORS connector is not available on Tasktop Cloud
IBM Engineering Requirements Management DOORS Next	6.0.6, 7.0, 7.0.1, 7.0.2
IBM Engineering Test Management	6.0.6, 7.0, 7.0.1, 7.0.2
IBM Engineering Workflow Management	6.0.6, 7.0, 7.0.1, 7.0.2






Jama Connect	8.42, 8.49, 8.56, 8.62, 8.66, 8.71, 8.74  Support All Patch Versions  Current On Demand (Cloud) Version
--------------	---












Micro Focus ALM /Quality Center	On Premise and SaaS versions:  12.60, 14.00-SaaS (Patch 0), 14.01 (SaaS), 15.0, 15.0.46, 15.5, 15.5.1, 16.00, 16.0.1
Micro Focus ALM Octane	12.60, 12.60.35, 12.60.47, 12.60.60, 15.0.20, 15.0.40, 15.0.46, 15.0.60, 15.1.20, 15.1.40, 15.1.60, 15.1.90, 16.0.100, 16.0.200, 16.0.300, 16.0.400  Current On Demand (Cloud) Version
Micro Focus Dimensions RM	12.8, 12.9  Support All Patch Versions

	Micro Focus PPM	9.4, 9.41, 9.42, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.62, 9.63, 9.64, 9.65, 9.66, 10.0  Support All Patch Versions
	Micro Focus Solutions Business Manager	11.5, 11.7  Support All Patch Versions
	Microsoft Azure DevOps Server	2017, 2017.1, 2017.2, 2017.3, 2017.3.1 2018, 2018.1, 2018.2, 2018.3, 2019 RC1, 2019, 2020, 2020.1  Support All Patch Versions  Current On Demand (Cloud) Version
	Microsoft Azure DevOps Services	Current On Demand (Cloud) Version*  Support All Patch Versions  *Please note limitations in <a href="#">Connector Documentation</a>
	Microsoft Project Server	2013 SP1, 2016*, 2019*, Project Online*  *Please note limitations in <a href="#">Connector Documentation</a>
	Microsoft SharePoint	2013 SP1, 2016, 2019, Sharepoint Online
	Microsoft Test Manager	Client Based Application accessing any supported version of Microsoft Team Foundation Server
	Modern Requirements4DevOps	

		Plug-in for all supported Microsoft Azure DevOps and Microsoft Azure DevOps Server versions
	Mozilla Bugzilla	5.0, 5.0.1, 5.0.2, 5.0.3, 5.0.4, 5.0.5, 5.0.6  Support All Patch Versions
	Pivotal Tracker	Current On Demand (Cloud) Version
	Planview AdaptiveWork (formerly Clarizen)	Current On Demand (Cloud) Version
	Planview AgilePlace (formerly LeanKit)	Current On Demand (Cloud) Version
	Planview Portfolios (formerly Enterprise One)	16 On Demand, 17 On Demand, 18 On Demand One  Support All Patch Versions
	Planview PPM Pro	Current On Demand (Cloud) Version
	Polarion ALM	19.0, 19.1, 19.2, 19.3, 20 R1, 20 R2, 21 R1, 21 R2, 22 R1  Support All Patch Versions
	PTC Windchill	11.1, 11.2, 11.2.1, 12.0, 12.0.1, 12.0.2
	PTC Windchill RV&S	12.3, 12.4, 12.5



		
	<p>Salesforce: Sales Cloud, Service Cloud, Marketing Cloud</p>	<p>Current On Demand (Cloud) Version</p>
	<p>ServiceNow: IT Service Management, IT Business Management (Agile Development/SDLC, PPM)</p>	<p>New York On Demand, Orlando On Demand, Paris On Demand, Quebec On Demand, Rome On Demand, San Diego On Demand</p>
	<p>ServiceNow Express</p>	<p>Current On Demand (Cloud) Version</p>
	<p>SmartBear QAComplete</p>	<p>11.2, 11.3, 11.4, 11.5, 11.6, 11.7 (for versions 11.7.1990 and later), 11.8, 11.9, 12.0, 12.1, 12.11, 12.12, 12.13, 12.14, 12.20, 12.21, 12.31, 12.40, 12.50, 12.60, 12.71, 12.80, 12.90, 14.0</p> <p>Current On Demand (Cloud) Version</p>
	<p>Sparx Systems Pro Cloud Server</p>	<p>4.0, 4.1, 4.2</p> <p>*Premium Editions only</p>

		
	Targetprocess	Current On Demand (Cloud) Version
	TestRail	7.0  Current On Demand (Cloud) Version
	Tricentis qTest	8.1.5, 8.4.4, 8.7.3, 9.0, 9.1.5, 9.3, 9.5.3, 9.6, 9.6.1, 9.7, 9.7.1, 9.7.11, 9.8.3, 10.2, 10.3, 10.4, 11.0  Support All Patch Versions  Current On Demand (Cloud) Version
	Tricentis Tosca	12.0, 12.1, 12.2, 12.3, 13.0, 13.1, 13.2, 13.3, 13.4, 14.0, 14.1, 14.2, 14.3, 15.0, 15.1, 15.2  Support All Patch Versions
	Trello	Current On Demand (Cloud) Version - Business Class Edition

	Whitehat Sentinel	Current On Demand (Cloud) Version
	Zendesk	Current On Demand (Cloud) Version
	Zephyr Squad for Jira	3.6, 4.0, 5.x, 6.x Current On Demand (Cloud) Version

# Repository Versions: End-of-Life Dates

## Tasktop Integration Hub 22.3 (July 19, 2022)

💡 For the following repository versions, you can request extended support until the End-of-Life date indicated. Otherwise, Tasktop will drop support for the repository version on the listed date.





If you are interested in extended support, please reach out to your [Tasktop contact](#).

### Repository Versions:

Release	End-of-Life Date
8.4	09 Sep 2022
8.5	21 Oct 2022
8.6	17 Dec 2022
8.7	17 Jan 2023
8.8	17 Jan 2023
8.9	17 Jan 2023
8.10	17 Jan 2023
8.11	17 Jan 2023
8.12	17 Jan 2023
8.13	17 Jan 2023
8.14	17 Jan 2023
8.15	02 Feb 2023

## Supported Repository Versions

💡 The following repository versions are currently supported in Tasktop Hub version 22.3.

	Repository	General Support (Tasktop 22.3)
	Aras Innovator	11.0 SP15
	Asana	Current On Demand (Cloud) Version
	Atlassian Jira Core	8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 9.0  Support All Patch Versions  Current On Demand (Cloud) Version
	Atlassian Jira Software	8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 9.0  Support All Patch Versions

8.16	23 Mar 2023
8.17	18 May 2023
8.18	01 Jul 2023
8.19	26 Aug 2023
8.20	19 Oct 2023
8.21	09 Dec 2023
8.22	16 Feb 2024





Release	End-of-Life Date
8.4	09 Sep 2022
8.5	21 Oct 2022
8.6	17 Dec 2022
8.7	17 Jan 2023
8.8	17 Jan 2023
8.9	17 Jan 2023
8.10	17 Jan 2023
8.11	17 Jan 2023
8.12	17 Jan 2023
8.13	17 Jan 2023
8.14	17 Jan 2023
8.15	02 Feb 2023
8.16	23 Mar 2023
8.17	18 May 2023
8.18	01 Jul 2023
8.19	26 Aug 2023
8.20	19 Oct 2023
8.21	09 Dec 2023

		Current On Demand (Cloud) Version
	Xray Test Management for Jira	6.2.1
	Atlassian Jira Service Management	4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 5.0  Support All Patch Versions
	Atlassian Jira Align	Current On Demand (Cloud) Version
	Blueprint	10.0, 10.1, 10.2, 10.3, 11.0, 11.1, 12.0, 12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7, 13.1, 13.2  4.0, 4.1, 4.2, 5.0, 5.1 (Storyteller)  Support All Patch Versions  Current On Demand (Cloud) Version  <b>Note:</b> With the release of Blueprint version 11.2, Blueprint Storyteller has been rebranded as Blueprint. Blueprint

8.22	16 Feb 2024
------	-------------

Release	End-of-Life Date
4.4	09 Sep 2022
4.5	17 Dec 2022
4.6	17 Dec 2022
4.7	17 Jan 2023
4.8	17 Jan 2023
4.9	17 Jan 2023
4.10	17 Jan 2023
4.11	17 Jan 2023
4.12	17 Jan 2023
4.13	17 Jan 2023
4.14	17 Jan 2023
4.15	02 Feb 2023
4.16	23 Mar 2023
4.17	18 May 2023
4.18	01 Jul 2023
4.19	26 Aug 2023
4.20	19 Oct 2023
4.21	09 Dec 2023
4.22	16 Feb 2024






Release	End-of-Life Date
10.0 4.0 (Storyteller)	17 Jun 2023
10.1	17 Jan 2023

		Storyteller versions 4.0 - 5.1 remain supported.
	BMC Remedy	9.1.03, 9.1.04, 18.05, 18.08, 19.02, 19.08, 19.11, 20.02  Support All Patch Versions
	Broadcom Clarity	15.7, 15.7.1, 15.8, 15.8.1, 15.9, 15.9.1, 16.0.0  Support All Patch Versions  Current On Demand (Cloud) Version
	Broadcom Rally	2018.1, 2.0, 2.1.0  Support All Patch Versions  Current On Demand (Cloud) Version
	Cherwell Service Management  (Only available for Planview OEM)	10.0.2
	codebeamer	8.2, 9.0, 9.1, 9.2, 9.3, 9.4, 9.5, 10.0, 10.1, 20.11, 21.04, 21.09

4.1 (Storyteller)	
10.2	17 Jan 2023
4.2 (Storyteller)	
10.3	17 Jan 2023
11.0	17 Jan 2023
5.0 (Storyteller)	
11.1	17 Jan 2023
5.1 (Storyteller)	

Release	End-of-Life Date
9.1.03	17 Jun 2023
9.1.04	17 Dec 2023
18.05	31 May 2023
18.08	31 Aug 2023
19.02	28 Feb 2024
19.08	22 Aug 2024
20.02	21 Feb 2025

Release	End-of-Life Date
15.7	30 Sep 2022
15.7.1	30 Sep 2022
15.8	17 Jan 2023
15.8.1	17 Jan 2023
15.9	30 Nov 2023
15.9.1	30 Nov 2023
15.9.2	30 Nov 2023
15.9.3	30 Nov 2023
16.0.0	30 Nov 2024



		Support All Patch Versions
	Digital.ai Agility	Enterprise and Ultimate: 21.0, 21.1, 21.2, 21.3, 22.0, 22.1  Support All Patch Versions  Current On Demand (Cloud) Version
	Digital.ai Release	8.0, 8.2, 8.5, 9.8, 10.0, 10.1, 10.2, 10.3
	Git	All  <b>*Note:</b> If using a supported Git Hosting Service, the version of the service used does not impact functionality. It is used to determine commit location.   Git connector is not available on Tasktop Cloud
	GitHub Issues	Enterprise 3.0 and prior, 3.1, 3.2, 3.3, 3.4  Current On Demand (Cloud) Version
	GitLab Issues	Enterprise and Community Edition:

Release	End-of-Life Date
2018.1	30 Jun 2023
2.0	30 Jun 2023
2.1.0	30 Jun 2023

Release	End-of-Life Date
8.2	01 Sep 2022
9.0	01 Sep 2022
9.1	01 Sep 2022
9.2	01 Sep 2022
9.3	01 Sep 2022
9.4	01 Sep 2022
9.5	01 Sep 2022
10.0	01 Sep 2022
10.1	01 Sep 2022
20.11	17 Jan 2023
22.04	30 Apr 2023
21.09	30 Sep 2023

Release	End-of-Life Date
21.0	17 Jan 2023
21.1	17 Jan 2023
21.2	17 Jan 2023

Release	End-of-Life Date
8.0	01 Sep 2022
8.2	01 Sep 2022
8.5	01 Sep 2022

		12.0, 12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7, 12.8, 12.9, 12.10, 13.0, 13.1, 13.2, 13.3, 13.4, 13.5, 13.6, 13.7, 13.8, 13.9, 13.10, 13.11, 13.12, 14.0, 14.1, 14.2, 14.3, 14.4, 14.5, 14.6, 14.7, 14.8, 14.9
		Support All Patch Versions
		Current On Demand (Cloud) Version
	IBM Rational ClearQuest	9.1.0
	IBM Engineering Requirements Management DOORS Family	9.5, 9.5.2, 9.6, 9.6.1, 9.7, 9.7.1, 9.7.2, 9.7.2.4  IBM DOORS connector is not available on Tasktop Cloud
	IBM Engineering Requirements Management DOORS Next	6.0.6, 7.0, 7.0.1, 7.0.2
	IBM Engineering Test Management	6.0.6, 7.0, 7.0.1, 7.0.2
	IBM Engineering Workflow	6.0.6, 7.0, 7.0.1, 7.0.2





9.8	17 Jan 2023
10.0	17 Jan 2023
10.1	17 Jan 2023
10.2	17 Jan 2023
10.3	17 Jan 2023

Release	End-of-Life Date
3.0 and prior	17 Jan 2023
3.1	17 Jan 2023
3.2	17 Jan 2023
3.3	17 Jan 2023
3.4	15 Mar 2023

Release	End-of-Life Date
12.0	17 Jan 2023
12.1	17 Jan 2023
12.2	17 Jan 2023
12.3	17 Jan 2023
12.4	17 Jan 2023
12.5	17 Jan 2023
12.6	17 Jan 2023
12.7	17 Jan 2023
12.8	17 Jan 2023
12.9	17 Jan 2023
12.10	17 Jan 2023

Release	End-of-Life Date
9.5	17 Jan 2023

	Management	
	Jama Connect	8.42, 8.49, 8.56, 8.62, 8.66, 8.71, 8.74  Support All Patch Versions  Current On Demand (Cloud) Version
	Micro Focus ALM /Quality Center	On Premise and SaaS versions:  12.60, 14.00-SaaS (Patch 0), 14.01 (SaaS), 15.0, 15.0.46, 15.5, 15.5.1, 16.00, 16.0.1
	Micro Focus ALM Octane	12.60, 12.60.35, 12.60.47, 12.60.60, 15.0.20, 15.0.40, 15.0.46, 15.0.60, 15.1.20, 15.1.40, 15.1.60, 15.1.90, 16.0.100, 16.0.200, 16.0.300, 16.0.400  Current On Demand (Cloud) Version
	Micro Focus Dimensions RM	12.8, 12.9  Support All Patch Versions
	Micro Focus PPM	9.4, 9.41, 9.42, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55,

9.5.2	17 Jan 2023
9.6	17 Jan 2023
9.6.1	17 Jan 2023

Release	End-of-Life Date
6.0.6	17 Jan 2023
7.0	07 Jul 2023
7.0.1	11 Dec 2023
7.0.2	11 Dec 2023

Release	End-of-Life Date
6.0.6	30 Apr 2023
7.0	07 Jul 2023
7.0.1	11 Dec 2023
7.0.2	11 Dec 2023

Release	End-of-Life Date
6.0.6	30 Apr 2023
7.0	07 Jul 2023
7.0.1	11 Dec 2023
7.0.2	11 Dec 2023

Release	End-of-Life Date
8.42	17 Jan 2023
8.49	17 Jan 2023





Release	End-of-Life Date
12.60	17 Jan 2023
15.0	31 Aug 2023

		9.62, 9.63, 9.64, 9.65, 9.66, 10.0
		Support All Patch Versions
	Micro Focus Solutions Business Manager	11.5, 11.7 Support All Patch Versions
	Microsoft Azure DevOps Server	2017, 2017.1, 2017.2, 2017.3, 2017.3.1 2018, 2018.1, 2018.2, 2018.3, 2019 RC1, 2019, 2020, 2020.1  Support All Patch Versions  Current On Demand (Cloud) Version
	Microsoft Azure DevOps Services	Current On Demand (Cloud) Version*  Support All Patch Versions  *Please note limitations in <a href="#">Connector Documentation</a>
	Microsoft Project Server	2013 SP1, 2016*, 2019*, Project Online*  *Please note limitations in <a href="#">Connector Documentation</a>
	Microsoft SharePoint	2013 SP1, 2016, 2019, Sharepoint

15.0.46	31 Aug 2023
15.5	30 Sep 2024
15.5.1	30 Sep 2024
16.0	31 Oct 2024
16.0.1	31 Oct 2024

Release	End-of-Life Date
12.60	17 Jan 2023
12.60.35	17 Jan 2023
12.60.47	17 Jan 2023
12.60.60	17 Jan 2023
15.0.20	31 Aug 2023
15.0.40	31 Aug 2023
15.0.46	31 Aug 2023
15.0.60	31 Aug 2023
15.1.20	31 Aug 2024
15.1.40	31 Aug 2024
15.1.60	31 Aug 2024
15.1.90	31 Dec 2024
16.0.100	31 Oct 2024
16.0.200	31 Oct 2024
16.0.300	31 Mar 2024
16.0.400	31 Oct 2024






Release	End-of-Life Date
12.8	17 Jan 2023
12.9	31 Jan 2024

		Online
	Microsoft Test Manager	Client Based Application accessing any supported version of Microsoft Team Foundation Server
	Modern Requirements 4DevOps	Plug-in for all supported Microsoft Azure DevOps and Microsoft Azure DevOps Server versions
	Mozilla Bugzilla	5.0, 5.0.1, 5.0.2, 5.0.3, 5.0.4, 5.0.5, 5.0.6  Support All Patch Versions
	Pivotal Tracker	Current On Demand (Cloud) Version
	Planview AdaptiveWork (formerly Clarizen)	Current On Demand (Cloud) Version
	Planview AgilePlace (formerly LeanKit)	Current On Demand (Cloud) Version
	Planview Portfolios (formerly Enterprise One)	16 On Demand, 17 On Demand, 18 On Demand  Support All Patch Versions

Release	End-of-Life Date
9.4	31 Oct 2022
9.41	31 Oct 2022
9.42	31 Oct 2022
9.50	17 Jan 2023
9.51	17 Jan 2023
9.52	17 Jan 2023
9.53	17 Jan 2023
9.54	17 Jan 2023
9.55	17 Jan 2023
9.62	31 Mar 2024
9.63	31 Mar 2024
9.64	31 Mar 2024
9.65	31 Mar 2024
9.66	31 Mar 2024

Release	End-of-Life Date
11.5	31 Dec 2022
11.7	17 Jan 2023

Release	End-of-Life Date
2017	17 Jan 2023
2017.1	17 Jan 2023
2017.2	17 Jan 2023
2017.3	17 Jan 2023
2017.3.1	17 Jan 2023
2018	17 Jan 2023
2018.1	17 Jan 2023

	Planview PPM Pro	Current On Demand (Cloud) Version
	Polarion ALM	19.0, 19.1, 19.2, 19.3, 20 R1, 20 R2, 21 R1, 21 R2, 22 R1  Support All Patch Versions
	PTC Windchill	11.1, 11.2, 11.2.1, 12.0, 12.0.1, 12.0.2
	PTC Windchill RV &S	12.3, 12.4, 12.5
	Salesforce: Sales Cloud, Service Cloud, Marketing Cloud	Current On Demand (Cloud) Version
	ServiceNow: IT Service Management, IT Business Management (Agile Development /SDLC, PPM)	New York On Demand, Orlando On Demand, Paris On Demand, Quebec On Demand, Rome On Demand, San Diego On Demand
	ServiceNow Express	Current On Demand (Cloud) Version
	SmartBear QAComplete	11.2, 11.3, 11.4, 11.5, 11.6, 11.7 (for versions 11.7.1990 and later), 11.8, 11.9, 12.0, 12.1,





2018.2	17 Jan 2023
2018.3	17 Jan 2023
2019	09 Apr 2024
2020	14 Oct 2025

Release	End-of-Life Date
19.0	01 Sep 2022
19.1	01 Sep 2022
19.2	01 Sep 2022
19.3	01 Sep 2022
20 R1	31 Dec 2022
20 R2	31 Dec 2022
21 R1	30 Apr 2023
21 R2	31 Oct 2023

Release	End-of-Life Date
11.1	01 Sep 2022
11.2	01 Sep 2022
11.2.1	01 Sep 2022
12.0	01 Sep 2022
12.0.1	30 Sep 2022
12.0.2	30 Jun 2024

Release	End-of-Life Date
12.3	31 Jan 2023

Release	End-of-Life Date
New York	17 Jan 2023
Orlando	17 Jan 2023

		12.11, 12.12, 12.13, 12.14, 12.20, 12.21, 12.31, 12.40, 12.50, 12.60, 12.71, 12.80, 12.90, 14.0  Current On Demand (Cloud) Version
	Sparx Systems Pro Cloud Server	4.0, 4.1, 4.2  *Premium Editions only
	Targetprocess	Current On Demand (Cloud) Version
	TestRail	7.0  Current On Demand (Cloud) Version
	Tricentis qTest	8.1.5, 8.4.4, 8.7.3, 9.0, 9.1.5, 9.3, 9.5.3, 9.6, 9.6.1, 9.7, 9.7.1, 9.7.11, 9.8.3, 10.2, 10.3, 10.4, 11.0  Support All Patch Versions  Current On Demand (Cloud) Version
	Tricentis Tosca	12.0, 12.1, 12.2, 12.3, 13.0, 13.1, 13.2, 13.3, 13.4, 14.0, 14.1, 14.2, 14.3, 15.0, 15.1, 15.2

Paris	17 Jan 2023
-------	-------------

Release	End-of-Life Date
8.1.5	01 Sep 2022
8.4.4	01 Sep 2022
8.7.3	01 Sep 2022
9.0	01 Sep 2022
9.1.5	01 Sep 2022
9.3	01 Nov 2022
9.5.3	17 Jan 2023
9.6	17 Jan 2023
9.6.1	17 Jan 2023
9.7	17 Jan 2023
9.7.1	17 Jan 2023
9.7.11	17 Jan 2023
9.8.3	17 Jan 2023
10.2	17 Jan 2023
10.3	17 Jan 2023
10.4	19 May 2024
11.0	29 May 2023

Release	End-of-Life Date
12.0	01 Nov 2022
12.1	17 Jan 2023
12.2	17 Jan 2023
12.3	17 Jan 2023
13.0	01 Sep 2022

		Support All Patch Versions
	Trello	Current On Demand (Cloud) Version - Business Class Edition
	Whitehat Sentinel	Current On Demand (Cloud) Version
	Zendesk	Current On Demand (Cloud) Version
	Zephyr Squad for Jira	3.6, 4.0, 5.x, 6.x Current On Demand (Cloud) Version

13.1	01 Feb 2023
13.2	05 May 2023
13.3	01 Jun 2023
13.4	22 Sep 2023
14.0	30 Nov 2023
14.1	01 Sep 2022
14.2	19 May 2024
14.3	01 Dec 2022
15.0	13 Dec 2024
15.1	17 Jan 2023
15.2	01 Jun 2025